# Security Report

Картина, която съдържа светлина

Описанието е генерирано автоматично

Content

[Security Report 1](#_Toc135917973)

[OWASP Top 10 Risks 3](#_Toc135917974)

[Reasoning 4](#_Toc135917975)

## OWASP Top 10 Risks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Likelihood | Impact | Risk | Actions possible | Planned |
| A01 Broken Access Control | Moderate | Severe | Moderate | Encrypting the parameters in the URL. | Yes |
| A02 Cryptographic Failures | Unlikely | Severe | Low |  |  |
| A03 Injection | Unlikely | Severe | Low |  |  |
| A04 Insecure Design | Unlikely | Moderate | Low |  |  |
| A05 Security Misconfiguration | Moderate | Low | Low | Error handling, delete console logs |  |
| A06 Vulnerable and Outdated Components | Unlikely | Low | Low |  |  |
| A07 Identification and Authentication Failures | Moderate | Moderate | Moderate | Include more validations and checks for registration |  |
| A08 Software and Data Integrity Failures | Unlikely | Low | Low |  |  |
| A09 Security Logging and Monitoring Failures | Moderate | Low | Low | Add more adequate logging events to the features |  |
| A10 Server-Side Request Forgery |  |  |  |  |  |

## Reasoning

***A01 Broken Access Control***: As a whole, my application is protected from users gaining privileges that they are not authorized to have. The only thing that needs to be fixed is to encrypt the URL parameters, so they wouldn’t be able to access the profile page, which is a big security risk, but I plan to fix it. Other than this flaw, however, there are other weak spots through which users can gain permissions they are not supposed to have.

***A02 Cryptographic Failures***: Encryption is used for the passwords which are the only truly sensitive part of the data the users need to fill in. They can’t be decrypted. I am using axios which encrypts the requests and makes them more secure.

***A03 Injection***: The SQL queries are using parameterized statements which means the risk of injection is fairly low.

***A04 Insecure Design***: I believe that Insecure Design is not a huge problem for my application because I have written a lot of tests which cover the code, the application uses the newest technologies as a whole, there is a segregation of layers and there is some validation for user input.

***A05 Insecure Design***: I believe the application is fairly secure in that regard because there are no unnecessary features, there is enough security hardening like security policies and the software is not out of date. I am going to look more into error handling and to ensure that there are no console logs which reveal information, but other than that, there aren’t any big issues.

***A06 Vulnerable and Outdated Components***: My application is not using outdated components and I delete all dependencies that are not being used, so I don’t believe there is an issue regarding this.

***A07 Identification and Authentication Failures***: There should be a bit more validation and checks if the password is strong enough by checking its length and complexity and testing it against a known list of weak passwords. However, the application doesn’t expose session identifier in the URL, the passwords are encrypted and in general, there aren’t many problems with identification and authentication.

***A08 Software and Data Integrity Failures***: I have Content Security Policies in place which minimize the risk of unneeded services having access to the application, the dependencies are not outdated, and I am not using untrusted URLs. There is a proper segregation and configuration of the CI/CD pipeline. Unsigned and unencrypted data is not sent to untrusted clients and there is a signature for the JWT tokens.

***A09 Security Logging and Monitoring Failures***: I will make sure to add more logging events which would be triggered upon error. However, that’s not one of my top priorities because there aren’t a lot of things on the application which are valuable since this is a web shop for videogames.

***A10 Server-Side Request Forgery***: There are Content Security Policies which decrease the attack surface of the application and only allow certain URL addresses for frontend and backend. There is also no feature in the application which allows regular customers to upload files which they could use to execute Request Forgery attacks. For those reasons, I believe the application is fairly secure against Request Forgery attacks.