OWASP Security Report

Individual Project Semester 3

Mihail Vasilev

Date: 16.12.2022

# Introduction

The purpose of this report is to analyze and outline any potential security risks that my application may be vulnerable to. I am going to take the OWASP top 10 and based on them I am going to determine whether the application is exposed to any possible threats. I am also going to state what actions I am going to take in order to solve the problems that have arisen.

# OWASP Top 10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Likelihood** | **Impact** | **Risk** | **Possible actions** | **Planned** |
| A01 – Broken Access Control | High | Severe | High | * Prevent unauthorized users from access * Different levels of permission depending on the type of user * Allow access to the API only from trusted origins | Yes |
| A02 – Cryptographic Failures | Moderate | Severe | Moderate | * All passwords are salted and hashed. | Yes |
| A03 - Injection | Unlikely | Severe | Low | * Make the application and the database protected from direct SQL injections with the use of Object Relational Mapping Tools (ORM). | Yes |
| A04 – Insecure Design | Low | Severe | Moderate | * The application follows the convention regarding designing the architecture of applications. | Yes |
| A05 – Security Misconfiguration | Moderate | Moderate | High | * The application is set with the required security values. * Any unnecessary features and test accounts will be removed upon completion. | Yes |
| A06 – Vulnerable and Outdated Components | Low | Moderate | Moderate | * The application is regularly updated and uses well maintained versions of software components. | Yes |
| A07 – Identification and Authentication Failures | High | High | High | * The application validates and does not allow users to create easy to crack authentication credentials. | Yes |
| A08 – Software and Data Integrity Failures | Low | Moderate | Low | * The application uses trusted libraries and component providers and the code is updated and reviewed regularly. | Yes |
| A09 – Security Logging and Monitoring Failures | Moderate | Moderate | Moderate | * A way of monitoring the applications activities should be implemented. | No |
| A10 – Server Side Request Forgery (SSRF) | Low | Low | Low | * The application sends verified data only to trusted URLs. | No |

# A01 – Broken Access Control

This a policy that ensures that users are not allowed to read and manipulate information outside of their intended permissions. In case of failure, data may be leaked, modified, destroyed or functions that otherwise are outside of the user’s range of permissions may be used.

To prevent from such attacks, it is advised to have your API endpoints protected from unauthorized users and untrusted origins (Cross-Origin Resource Sharing). Another example on how to prevent would be to have the JWT tokens short-lived so that a potential attacker has a small window of opportunity.

# A02 – Cryptographic Failures

Depending on the information transferred and stored different protections methods must be implemented in order to comply with regulations such as EU's General Data Protection Regulation (GDPR). Sensitive information such as passwords, credit card numbers and more need to be stored securely. In general, the less sensitive information you store the better. Any data that is unnecessary must not be stored. Passwords need to be hashed and salted with the use of strong and adaptable algorithms. Deprecated and outdated technologies should be avoided in order to keep the information as secured as possible.

# A03 – Injection

The application can become vulnerable if improper commands to the database are used, object-relational mapping is not properly implemented and the data received from outside is not properly validated, filtered and sanitized by the application. Some of the most common injections are SQL, NoSQL, OS command and others. To prevent from such attacks the application should use a safe API. All outside data should be validated before it is used. The access to the database should be strictly controlled.

# A04 – Insecure Design

The application must be design in such a way that it is secured from potential attacks. The idea is to create robust code that is tested against the know dangers and attack methods that may be used against it. A well-designed application with good integrity and proper user story developed workflow and failure states ensures that the information is kept safe and misuse of data is not permitted. These factors must be taken into account when designing the application since it is not a separate part that can be added at a later stage but the application’s structure in general.

# A05 – Security Misconfiguration

The application can be prone to such misconfigurations in cases where appropriate security measures are not implemented or where test and default accounts are kept enabled or unchanged. Error handling may reveal overly informative information to users who are not supposed to see these kinds of data. The application frameworks such as Spring or ASP.NET are left with improper security settings. Any unnecessary features that were used during development are left in the final product. To prevent such issues, it is advised to keep all security configuration values up to date and secure as well as keep the application clean from any unnecessary features and components and properly configure the application for its actual users.

# A06 – Vulnerable and Outdated Components

The application may be vulnerable if it uses outdated components or is unsupported by the developers. Regular updating and monitoring should be implemented to ensure that the application is up to date. Components must be obtained from official and trusted sources. Any unused and unnecessary dependencies should be removed.

# A07 – Identification and Authentication Failures

Applications with poor user authentications procedures are prone to identification and authentication failures. Weaknesses such as permitting automated and brute force attacks, weak or default passwords, ineffective and weak credential recovery processes or improper usage session identifiers can lead to breaches. To prevent from such potential problems, ensure that the users are not allowed to create simple and easy to crack authentication credentials and implement multi-factor authentication. The application must not be deployed with any default admin user credentials.

# A08 – Software and Data Integrity Failures

Applications that rely on many outside plugins, libraries and modules especially from untrusted sources are prone to integrity vulnerabilities. Sufficient verification must be implemented when auto-updated and use only trusted sources. Make sure that any potential for introducing malicious software is kept to a minimum by always having code review before deployment.

# A09 – Security Logging and Monitoring Failures

Applications need to have sufficient logging, detection and monitoring to ensure that any security breaches are caught as soon as possible. The application needs to have a way of detecting and alerting in case of attacks as well as keep a log of the activities happening. This way any suspicious processes can be detected and stop at early stages and prevent data leaks.

# A10 – Server-Side Request Forgery (SSRF)

When a remote recourse is fetched to a user without properly validating the user’s URL SSRF attacks can occur. The attacker makes so that the application sends the information to an unexpected destination which otherwise would not be permitted. To prevent such vulnerabilities, it is best that the application validates and sends only to trusted and destinations only on the allowed list. HTTP redirections are disabled and raw responses are never sent directly to the clients.

# Conclusion

In conclusion, the application that I am making is well maintained, has proper design architecture and any outside vulnerabilities are kept to a minimum. Of course, there is always a lot of things that can be done to make the application even more secured and in the next stages of development I will strive to make it even better.