|  |
| --- |
| **Fontys – University of Applied Sciences - ICT** |
| Individual track project: Online inventory system |
| Integration of OWASP Principles into Individual Project |

|  |
| --- |
| Martin Georgiev  Student number: 3782484  Eindhoven, Friday, November 27, 2020 |

Document Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| *Date* | *Version* | *Author* | *Comments* |
| 27-11-2020 | 1.0 | **Martin Georgiev** | Added the initial body structure and text of the OWASP report |
|  |  |  |  |

Table of contents

[1 Introduction 4](#_Toc57407999)

[1.1 Document Purpose 4](#_Toc57408000)

[1.2 Document Overview 4](#_Toc57408001)

[2 Meeting the criteria 5](#_Toc57408002)

[2.1 What is OWASP? 5](#_Toc57408003)

[2.2 The Top Security Principles 5](#_Toc57408004)

[3 References 6](#_Toc57408005)

# Introduction

## Document Purpose

The purpose of this report is to provide a short and concise overview and analysis of the security aspect of the project and the integration process of the OWASP principles into it. The goal is to provide more information on development, including any challenges and decisions that had to be made and any other piece of data that had to be considered to deliver a software product that complies with the abovementioned standards.

## Document Overview

This document will cover the aforementioned information in the following sections, considered important to all partaking stakeholders (in this case – teacher/mentor and student). Namely, these sections are:

* **Meeting the criteria:** An in-depth look at the OWASP principles and the way they correlate to the project. If the project applies any of the technologies described in these standards, a description will be given that explains the way it was implemented and the decisions behind it.

# Meeting the criteria

## What is OWASP?

The Open Web Application Security Project, or also widely known as OWASP, is an international organization that is dedicated to establishing web application security standards. One of the foundation’s core principles is to freely share all their materials on their platform to everyone who desires to improve their software application security.

The OWASP Top 10 is a report that is put together by a team of security experts from all over the world. It outlines all web security concerns that a developer might come across during development. It is also referred to as an awareness document by the foundation. People are incentivized to incorporate the information provided in the report into their processes to minimize and/or completely mitigate any potential security risks. (Cloudflare, n.d.)

## The Top Security Principles

### Injection

This section refers to injection flaws that could result in attacks to systems such as SQL, NoSQL, OS, LDAP, etc. As a result of this attack, untrusted data is sent to an interpreter as part of a query or a command. Injection attacks can be prevented through the validation and sanitization of any user-submitted data.

The backend side of the full-stack web application of the individual assignment handles communication between the RESTful API web service and the database structure. That is why it is not wrong to assume that this security risk solely affects the backend of the project. The measure that was taken to resolve this issue is the use of prepared statements with parameterized queries in order to eliminate the possibility for SQL injections.

### Broken Authentication

Any vulnerabilities in the authentication system can give an opportunity for attacks to access sensitive user data with the chance to compromise the entire system if not handled correctly. During the process, the attacker can potentially compromise user passwords, keys, session tokens and any other flaws to temporarily or permanently identify themselves as a user that has access to restricted content.

The way this issue is handled in the individual project is the use of an encrypted authentication system that utilizes session tokens as a secondary meaning of identification and an additional authorization system that prevents any authenticated users from accessing backend API endpoints that are meant to be used only specific roles.

### Sensitive Data Exposure

Sensitive data must be protected to mitigate a data exposure risk through means such as encryption as well as disabling of caching of any sensitive information.

For the individual project, all backend API endpoints are thoroughly reviewed not to expose sensitive user data that is not used directly by any of the frontend processes. In addition to that, an ORM database system is used with plans to encrypt database data.

### XML External Entities (XEE)

This type of security vulnerability affects web applications that parse Extensible Markup Language (**XML**) input. This input can potentially reference an external entity (any sort of digital storage unit) that can be exploited through the parser. The XML parser can be duped into sending data to an unauthorized external entity, which can send sensitive user information directly to an attacker.

One of the ways to prevent this type of attack is one that the backend system utilizes heavily – the use of less complex data structures, as JavaScript Object Notation (**JSON**), to transfer information between the two systems of the full-stack application.

### Broken Access Control

Access control refers to any system that controls access to information or functionality. With broken access controls, authorization could be bypassed and role restricted functions executed as if the attack were that privileged user (e.g. administrator accounts). For instance, simply changing the web URL while the account is logged in could result in access to content if the web application does not have any additional authorization controls.

The individual project attempts to solve this particular vulnerability by implementing authentication/authorization tokens. Additionally, additional verification methods are added to the frontend side of the web application to prevent users from accessing sensitive data through the abovementioned attack method.

### Security Misconfiguration

Security misconfiguration is a common type of vulnerability that is often the result of using default configuration on systems related to security. Additionally, displaying overly descriptive errors to the end user could also reveal any additional vulnerabilities. As mentioned by the OWASP report, removing any unused features in the code ensures that errors are more generalized and fewer sources can lead to errors.

There is no guarantee that this has been correctly executed for the individual project. If any misconfigurations become apparent, they will be promptly fixed. Moreover, code will be cleaned up prior to deployment to production.

### Cross-Site Scripting

Cross-site scripting (**XSS**) flaws occur when web applications include untrusted data into a web page without proper validation or escaping. As a result, the attacker could potentially create an HTML or JavaScript file through a browser’s API and use it to execute scripts on the victim’s browser. This could result in user session hijacks or user redirection to malicious sites through the web application’s interface.

This issue affects the frontend of the project. Fortunately, the framework that is used to develop the web application (ReactJS) provides some built-in cross-site scripting protection.

### Insecure Deserialization

Insecure Deserialization targets any web application that serializes and deserializes data. Data could be potentially tampered with upon deserialization if it is done insecurely. Insecure deserialization exploit can come from untrusted sources and could result in serious consequences like DDoS attacks and code execution attacks. The only way to mitigate this vulnerability is to prohibit data deserialization from untrusted sources. This process can also be monitored as an attempt to catch attackers but it will not fix the core issue. (Cloudflare, n.d.)

This security risk does not affect the individual project as no known serialization and deserialization is done that is not part of the framework.

### Using Components with Known Vulnerabilities

Any components (libraries, frameworks and any other software modules) that run with the same privileges as the main application could be compromised and used to exploit the web application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. (The OWASP Foundation, n.d.)

There is no guaranteed way to resolve this vulnerability. Fortunately, the individual project relies on popular frameworks and libraries that have been well tested and are relatively secure.

### Insufficient Logging and Monitoring

Insufficient logging and monitoring could lead to data breaches that go undetected by any participating internal parties related to the development of the software. *In fact, the average discovery time for a breach is around 200 days after it has happened (The OWASP Foundation, n.d.)*. Implementing logging and monitoring systems could help to respond to such incidents in a timely manner.

The current implementation of the individual assignment has no logging or monitoring systems. Due to the nature of the project, such a system might not be developed at all. Nevertheless, this point will be taken into consideration for any future tasks.

# References

Cloudflare. (n.d.). *What is OWASP? What Are The OWASP Top 10?* Retrieved from Cloudflare: https://www.cloudflare.com/learning/security/threats/owasp-top-10/

The OWASP Foundation. (n.d.). *Top 10 Web Application Security Risks*. Retrieved from OWASP: https://owasp.org/www-project-top-ten/