Security - Owasp

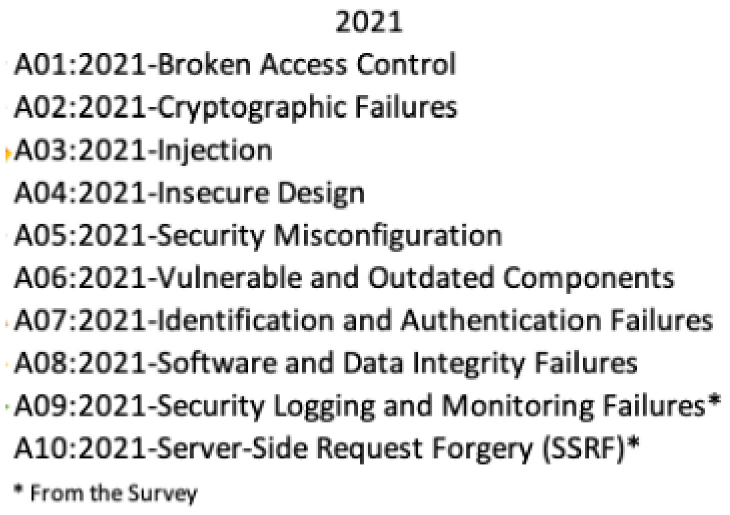
This document is intended to detail what OWASP is and some security concerns that could be important. I also want to describe what security measures I already took.

OWASP

What is OWASP:
OWASP (Open Web Application Security Project) is an open community dedicated to improving the security of software applications. It provides resources, guidelines, and tools for developers, security professionals, and organizations to enhance the security of web and mobile applications.

OWASP is best known for the OWASP Top Ten, a regularly updated list of the most critical web application security risks. The OWASP Top Ten provides an awareness of common vulnerabilities and helps prioritize security efforts during application development.

The top 10 is as follows:



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A06:2021-Vulnerable and Outdated Components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 (SSRF)*
* From the Survey

How do these concerns influence my application?

Broken access control

This means that people that shouldn’t have access to your systems have access. This is a big potential problem as this could give people access to abilities that can break your application. This can also cause databreaches which are a big issue.

The solution to this problem is to setup proper authentication, limit actions based on roles, limit access to what is strictly neccessary and to only store what is absolutely needed.

Personally I restricted endpoints that shouldn’t be exposed, created authentication based on roles and tried to limit unneccessary data being transported.

Cryptographic Failures

This means that encrypted data gets exposed. This can be due to a weak encryption key, old hashing methods or insecure communication methods.

The solution is to use a secured communication methods such as TLS, use current encryption methods and to use a strong encryption key.

Personally I use an up to date encryption methods with a non-default key.

Injection

This means that code controlled by a third party can run in or on your systems. This can cause all manner of bad things.

The solution is to escape most variables, sanitise all input and prevent code being injected into parts of your code that have wide reaching authority.

Personally I do not run a lot of risk here as my input gets sanitised or does not have wide reaching access to the rest of the program.

Insecure Design

This is a broad category. Basically this is here to alert developers that security is not a problem with a magic fix-all solution. Security is a constant progress more akin to a philosophy or a continuous process.

The solution here is to constantly think about what security tools might be needed and implement the proper defenses throughout the development.

Security Misconfiguration

This is about accidentaly exposing credentials and leaving the proverbial door wide open. This can also include using standard or default credentials which allow unauthorised parties to find their way into systems.

Personally this is a decent risk, I took good care to remove credentials from my code and any online repositories. Any credential that was previously exposed got changed so is no longer accurate. However my cloud is mostly a default setup which could be exploited.

Vulnerable and outdated components

This means having outdated dependencies that are being used in your program. Especially nowadays it is very easy to use a lot of different dependencies and it becomes difficult to verify their security and thus your programs security.

Personally I mostly use popular dependencies so I hope to mitigate this risk by having well known and active code being used. But this remains a real risk and static code analysis is a good way to mitigate it.

Identification and Authentication Failures

This means that someone could bruteforce their way through the authentication layers.

Personally my application runs a decent risk here as I currently do not enforce certain password standards. Nor do I use a multi-factor authentication system.

Software and Data integrity Failures

A third party dependancy could become compromised and thus compromise your security aswel.

To mitigate this I do static code analysis and on occassion do a dependancy check. However this is not part of my pipeline and therefore vulnerable to human failure.

Security logging and monitoring failures

This means that without properly monitored code small errors could cause great problems and open up security issues of which you are not aware.

Personally I have written tests and setup monitoring systems. However this is currently still too limited and should be expanded on.

Server-Side request Forgery

This means that people can misdirect the server into making a request or retrieving data from an unauthorized source. Think code injection but then from the server side instead of client side.

The solution is to enforce URL schemes, strict firewall policies and even VPN connections. Essentially restrict the network and how far it can reach.

Security measures I took

1. SonarCloud code analysis
2. Solved SonarCloud security hotspots.
3. Hid credentials from code through the use of an .env file.
4. Hid credentials in pipelines through the use of github secrets.
5. Used an IDE plugin for code quality.