**Security OWASP research**

**Student:** Jorn Kersten

**Student number:** 483331

**Class:** DB-S3-DB03

Table of contents

[Introduction 3](#_Toc121141271)

[OWASP top 10 4](#_Toc121141272)

[1.1 A01:2021-Broken Access Control 4](#_Toc121141273)

[1.2 A02:2021-Cryptographic Failures 4](#_Toc121141274)

[1.3 A03:2021-Injection 4](#_Toc121141275)

[1.4 A04:2021-Insecure Design 5](#_Toc121141276)

[1.5 A05:2021-Security Misconfiguration 5](#_Toc121141277)

[1.6 A06:2021-Vulnerable and Outdated Components 5](#_Toc121141278)

[1.7 A07:2021-Identification and Authentication Failures 5](#_Toc121141279)

[1.8 A08:2021-Software and Data Integrity Failures 6](#_Toc121141280)

[1.9 A09:2021-Security Logging and Monitoring Failures 6](#_Toc121141281)

[1.10 A10:2021-Server-Side Request Forgery 6](#_Toc121141282)

# Introduction

~~This document was created to keep track of my researches on my software characterisation. By this I mean the researches about which front-end frameworks I can use, the programming languages I can use for my back-end. But also what type of database I want to use, and in addition what kind of database I want to use then.~~

The structure of this document is as follows. The first chapter is my research on my front-end framework. This research starts with the popularity between the three frameworks I have chosen. I then list all the pros and cons for each framework. All this is followed by a conclusion in which I make my choice and substantiate it.

Next, I research the back-end programming language, the structure of which is slightly different. I first tell something about the languages, with both a short description followed by the features they have. Then I look at the differences between the two, followed by the conclusion. In it, I make my choice and substantiate it.

Finally, I compare database types, writing pros and cons for each type. After this comes the conclusion in which I make my choice, and again I substantiate my choice. The next chapter compares three databases I could use for my application. Here I again talk briefly about the database, then I write down the pros and cons for each database. This is followed by a conclusion. In it, I make a choice based on my project and justify the choice.

# OWASP top 10

The OWASP top 10 is a list of the top 10 most common security mistakes in web applications. As such, these errors are often mistakes that could have been prevented.

You can use the OWASP top 10 before you start programming to find out what to look out for, but also to identify errors you already have in your application.

## A01:2021-Broken Access Control

Access control ensures that a user can make changes only within their scope. By this I mean that only when the user is logged in is he allowed to modify his own e-mail address, for example. So what often goes wrong when access control is broken is that a user can modify, delete, or manipulate the data of an account outside his own.

Some examples are:

* When functions or other data access points are not checked for roles. So that anyone can access where data can be modified.
* Passing access control checks in order to manipulate data, delete a post or another user.

## A02:2021-Cryptographic Failures

Cryptographic failure involves not thinking about what data should be secured. For example, passwords, credit card numbers, health records, personal information, etc. This refers to anything covered by privacy laws.

Common problems then are:

* When data is sent in plain text, the data is then not secured when it goes over the internet to the user. All the people who intercept the message then can easily read what is sent. Protocols to avoid this are to send the data via: HTTPS, SMTP or FTP. These protocols provide a secure connection from which it is not possible to retrieve what is in the sent message.
* If important data such as passwords is not encrypted when a message is sent between server and user, it can be intercepted and read. Also, within the server, it is not known what it should actually be which makes it secure within the server.

## A03:2021-Injection

Injection means that a user can inject data into the database by sending an SQL command, for example. This allows data to be manipulated or deleted on a database, this can cause the entire database to be broken. Data injection is often done when a form can be filled in, here all kinds of data can be entered and then sent to the database.

The problems this can cause are:

* If data entered by a user is not validated, filtered or cleaned. Then the data as submitted is sent directly to the database. A command can then be executed in the database and thus break the database.

## A04:2021-Insecure Design

Insecure design is not the source of the top 10 risk categories. There is a difference between insecure design and insecure implementation. Often, the difference between the two can be found at the source of both problems. Insecure implementation is the failure to properly implement the engineered solution, while insecure design often does not properly consider how data can be stored or processed securely.

Examples of insecure design can be as follows:

* No (proper) unit tests are written to validate all critical flows.
* Correct threat models are not properly followed. These models are for authentication, access control and business logic.

## A05:2021-Security Misconfiguration

Security misconfiguration often involves mistakes when enabling or adjusting settings. Examples include setting up ports for certain web pages and then forgetting to close them, turning on unnecessary settings for testing and forgetting to turn them off again, not changing default passwords to a more secure password, etc. This list of problems is therefore often where things go wrong.

## A06:2021-Vulnerable and Outdated Components

The vulnerability of an application depends on many components within an application. When an application is not kept up to date by a programmer, if at all, it quickly becomes dated. When software or dependencies are not maintained or updated, security problems can arise. In this way, the application becomes very susceptible to hackers and other malicious programs.

Common examples of vulnerable applications are:

* When the software, dependencies and other important things for a programme are not maintained or updated.
* When the software is not regularly get checked for vulnerability by yourself or an external party.
* When updated libraries are not tested to see if they can still work safely within the software or other parts of the software.

## A07:2021-Identification and Authentication Failures

Identification and Authentication Failures occur in applications that do nothing to authenticate user actions. Another problem can occur when incorrect or bad passwords are allowed with an account. These are passwords like "Pasword1" or "admin/admin". If applications do not hash the password when saving or sending over from the database to the website to compare passwords. A final common error is with the session id. Sometimes it is included in a url, or the session is not validated correctly.

## A08:2021-Software and Data Integrity Failures

Software and data integrity failures are common in applications that rely heavily on plugins, libraries, or modules from untrusted sources. Because these components were not created by the developer of the application, this is where things can often go wrong. External components are not always updated much, in addition, the updates are not always properly maintained by developers. Many applications therefore use automatic updates, malicious applications can take advantage of this by performing bad updates to destroy applications.

## A09:2021-Security Logging and Monitoring Failures

If there are breaches in a system, they are hard to get through. Something that helps with that is logging items when something is done within a system. To make this clear, you could log when logins are made, if a login is often failed. It is also important then to use clear logs. When it is not possible to get what is there directly from the log, it becomes very difficult to figure out what is going on.

## A10:2021-Server-Side Request Forgery