Security Semester 6

Naam: Rienk Engbrenghof

Studentnummer: 429059

Klas: S6-RB03T

# Introductie

In dit document zullen de keuzes rondom security gedocumenteerd worden. Dit wordt voornamelijk gedaan op basis van de OWASP top 10.

# Versiebeheer

|  |  |
| --- | --- |
| 5/9/2022 | Security document aangemaakt en aanpassingen gemaakt. |
| 1/6/2022 | OWASP punten uitgewerkt. |

# Repositories

Link naar repository: https://github.com/Pretter-S6

# Inhoudsopgave

[1. OWASP top 10 5](#_Toc102984092)

[1.1 Broken Access Control 5](#_Toc102984093)

[1.2 Cryptographic Failures 5](#_Toc102984094)

[1.3 Injection 5](#_Toc102984095)

[1.4 Insecure Design 5](#_Toc102984096)

[1.5 Security Misconfiguration 5](#_Toc102984097)

[1.6 Vulnerable and Outdated Components 5](#_Toc102984098)

[1.7 Identification and Authentication Failures 5](#_Toc102984099)

[1.8 Software and Data Integrity Failures 5](#_Toc102984100)

[1.9 Security Logging and Monitoring Failures 5](#_Toc102984101)

[1.10 Server-Side Request Forgery 5](#_Toc102984102)

[2. Conclusie 6](#_Toc102984103)

# OWASP top 10

## 1.1 **[Broken Access Control](https://owasp.org/Top10/A01_2021-Broken_Access_Control/)**

**Beschrijving**

Dit punt van de OWASP top 10 zegt dat gebruikers niet kunnen handelen buiten de dingen die zij mogen doen. Wanneer dit niet goed gehandhaafd wordt, kan er wellicht informatie worden aangepast, ingezien of verwijderd.

**Implementatie**

Om te voorkomen dat dit gebeurt bij Pretter, heb ik ingebouwd dat wanneer er genavigeerd wordt naar een bepaalde url. Er altijd gekeken wordt of de gebruiker is ingelogd. Wanneer dit niet het geval is dan wordt de gebruiker teruggestuurd naar de loginpagina.

Ook de verschillende http methodes worden beveiligd. Dit omdat er maar 1 point of attack is en dat is de API gateway. Omdat dit de enige aanspraak is, hoeft alleen hier beveiligd te worden dat ongeautoriseerde gebruikers niks kunnen.

In mijn applicatie zijn cookies op de gedocumenteerde wijze toegepast, echter sluit dit niet uit dat er eventueel iets mis mee zou kunne gaan.

## 1.2 [**Cryptographic Failures**](https://owasp.org/Top10/A02_2021-Cryptographic_Failures/)

**Beschrijving**

Dit onderwerp stond voorheen bekend als sensitive data exposure, dus voor dit punt moet er worden nagedacht over hoe data moet worden beveiligd. Hierbij moet ook worden gedacht aan data die valt onder bepaalde wetten zoals de AVG.

**Implementatie**

Er is gebruik gemaakt van het entity framework, dit framework zorgt al voor een geavanceerde en vernieuwde manier van data opslaan en ophalen. Ook wordt er gebruik gemaakt van de nieuwste MSSQL server, dus de data wordt passief ook juist opgeslagen, zonder eigen geschreven query’s.

Voor wachtwoorden wordt in de applicatie gebruik gemaakt van encryption, dit voor wanneer er mogelijk data wordt uitgelezen. Dan is deze op z’n minst encrypted.

## 1.3 [**Injection**](https://owasp.org/Top10/A03_2021-Injection/)

**Beschrijving**

Injection kan plaatsvinden wanneer een applicatie niet controleert op user-input. Hierbij zou een parameter in de query kunnen komen te staan, die ervoor zorgt dat alle data verwijderd wordt. Op deze manier zou ook data kunnen worden opgehaald door middel van user-input.

**Implementatie**

Het entity framework voorkomt voor een groot deel al dat er injection mogelijk is via user-input. Het maakt gebruik van LINQ-to-Entity queries en staat het toe om je eigen queries te schrijven. Alleen dit heb ik niet gedaan binnen mijn applicatie.

Binnen de applicatie worden ook geen onnodig lange chars toegestaan, hierdoor verklein ik de kans dat er queries geschreven worden binnen de user-input.

## 1.4 [**Insecure Design**](https://owasp.org/Top10/A04_2021-Insecure_Design/)

**Beschrijving**

Insecure design omschrijft dat de security vanaf het design goed geïmplementeerd moet worden. Dit verschilt enorm van een insecure implementation. Wanneer je een goed design opzet, kan je applicatie worden geprogrammeerd met de juiste security waarborging.

**Implementatie**

Voor mijn applicatie heb ik gebruik gemaakt van een DevSecOps, waarin de code in de pipeline constant gecontroleerd wordt op eventuele security risks. Dit wordt gedaan door sonarcloud. Ook heb ik op aanraden user stories opgesteld omtrent de verschillende security risks. Zodat wanneer je deze doorloopt, je voldoet aan bepaalde security eisen.

Ook heb ik voor de applicatie unit-testen geschreven, deze controleren de use-cases. En of de flow van de applicatie juist gebruikt wordt. Ook de architectuur is van groot belang, ik heb gekozen voor een microservice architectuur, hierin zijn maar bepaalde lagen aanspreekbaar.

## 1.5 [**Security Misconfiguration**](https://owasp.org/Top10/A05_2021-Security_Misconfiguration/)

**Beschrijving**

Dit onderwerp omschrijft vooral onnodige configuraties, die niet aan zijn gezet of niet uit zijn gezet. Waardoor er security risks onstaan. Een voorbeeld hiervan is poorten op een service open zetten, die niet worden gebruikt. Hierdoor wordt de attack surface vergroot.

**Implementatie**

In mijn applicatie zijn alleen de poorten opengezet die daadwerkelijk worden gebruikt, het is maar mogelijk om via 1 poort te communiceren. Ook zijn in de database alle standaard wachtwoorden uitgeschakeld.

Error messages worden intern afgehandeld, deze worden nooit terug gelogd aan de gebruiker. Sonarcloud geeft ook meldingen wanneer er componenten of stukken code worden gebruikt, die onnodig zijn. Deze haal ik vervolgens ook weg, om de security te verbeteren.

## 1.6 [**Vulnerable and Outdated Components**](https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/)

**Beschrijving**

Het is van belang om als developer de versies van je applicatie te weten, zodat je weet of ze up-to-date zijn of niet. Hoe ouder de software, hoe groter de kans dat er risico’s in zitten. Hierbij moet ook de compatibiliteit goed overwogen worden.

**Impelementatie**

In mijn applicatie wordt van de frameworks de nieuwste versie gebruikt. Ook de database maakt gebruik van de nieuwste MSSQL server versie.

Sonarcloud controleert mijn applicatie bij elke push op eventuele outdated software of dependencies. Hierbij waarschuwt sonarcloud voor eventuele security risks.

## 1.7 [**Identification and Authentication Failures**](https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/)

**Beschrijving**

Dit onderwerp stond eerst bekend als broken authentication. Het omschrijft het juist authenticeren van gebruikers. Hierbij zijn er verschillende aanvallen die ervoor kunne zorgen dat de inlog kan worden omzeild.

**Implementatie**

In mijn applicatie zijn zoals gezegd de standaard wachtwoorden niet mogelijk om te gebruiken. Ook is de data volgens de Cryptographic failures, juist opgeslagen en encrypted. De session ID wordt nooit getoond in de URL.

Ter versterking zou ik nog in de applicatie kunnen bouwen dat het wachtwoord x aantal karakters lang moet zijn en speciale tekens moet bevatten.

## 1.8 [**Software and Data Integrity Failures**](https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/)

**Beschrijving**

Dit onderwerp omschrijft vooral projecten waarin de infrastructuur rondom de code niet goed is opgezet. Hierbij kunnen applicaties gebruik maken van niet betrouwbare sources, repositories of netwerken. Hierbij kan een insecure CI/CD zorgen voor risico’s.

**Implementatie**

Mijn applicatie maakt alleen gebruik van betrouwbare externe partijen. Zoals sonarcloud en docker. Ook worde er geen automatische updates binnen de applicatie gedaan, hierdoor moet er handmatig worden gekeken of de versies goed zijn.

De CI/CD pipeline is correct opgezet en hierin zijn geen onbetrouwbare partijen gebruikt, ook wordt de code niet buiten GitHub gepublisht. Hierbij wordt geen data buiten GitHub geëxporteerd.

## 1.9 [**Security Logging and Monitoring Failures**](https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/)

**Beschrijving**

Dit onderwerp gaat over dat een applicatie moet loggen en monitoren wanneer er iets fout gaat. Hierbij kan gedacht worden aan het loggen van een uitzonderlijk hoge aantal keer inloggen. Waarschuwingen die duidelijk zijn en gelogd worden en deze logs worden ergens opgeslagen.

**Implementatie**

Momenteel worden er geen errors of waarschuwingen in mijn applicatie gelogd. Het enige is dat sonarcloud security logt. Dit zou wel nog slim zijn om in mijn applicatie te bouwen, zodat ik een overzichtelijke log heb van eventuele risico’s. Hierdoor zou ik sneller kunnen handelen, en gericht het security risico kunnen oplossen.

## 1.10 [**Server-Side Request Forgery**](https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/)

**Beschrijving**

Dit onderwerp omschrijft dat er calls kunnen worden gedaan naar bepaalde enpoints, en dat de server kant niet valideert van wie deze call afkomt. Hierdoor zou iedereen een call kunnen doen zonder geautoriseerd te zijn.

**Implementatie**

Momenteel bevat mijn applicatie dit nog niet, het zou wel slim zijn om te implementeren. Het zou voornamelijk effectief zijn om het te implementeren op de API-Gateway. Omdat hier de attack surface het groots is.

# Conclusie

Na het doorlopen van de OWASP top 10, is naar voren gekomen dat mijn applicatie op veel punten beveiligd is.

Het zou echter nog beter kunnen bij de laatste twee punten, wanneer ik genoeg tijd heb om dit te implementeren dan zal ik deze punten verder uitwerken.