OWASP

Inhoud

[Injection 1](#_Toc60844577)

[Wat is het? 1](#_Toc60844578)

[Hoe komt her voor in mijn project? 1](#_Toc60844579)

[Hoe voorkom ik het? 1](#_Toc60844580)

[Broken Authentication 1](#_Toc60844581)

[Wat houd het in 1](#_Toc60844582)

[Hoe komt her voor in mijn project? 1](#_Toc60844583)

[Hoe voorkom ik het? 1](#_Toc60844584)

[Sensitive Data Exposure 2](#_Toc60844585)

[Wat is het? 2](#_Toc60844586)

[Hoe voorkom ik het? 2](#_Toc60844587)

[Hoe komt her voor in mijn project? 2](#_Toc60844588)

[XML External Entities (XXE) 2](#_Toc60844589)

[Wat is het? 2](#_Toc60844590)

[Hoe komt her voor in mijn project? 2](#_Toc60844591)

[Hoe voorkom ik het? 2](#_Toc60844592)

[Broken Access Control 2](#_Toc60844593)

[Wat is het? 2](#_Toc60844594)

[Hoe komt her voor in mijn project? 2](#_Toc60844595)

[Hoe voorkom ik het? 2](#_Toc60844596)

[Security Misconfiguration 3](#_Toc60844597)

[Wat is het? 3](#_Toc60844598)

[Hoe komt her voor in mijn project? 3](#_Toc60844599)

[Hoe voorkom ik het? 3](#_Toc60844600)

[Cross-Site Scripting XSS 3](#_Toc60844601)

[Wat is het? 3](#_Toc60844602)

[Hoe komt her voor in mijn project? 3](#_Toc60844603)

[Hoe voorkom ik het? 3](#_Toc60844604)

[Insecure Deserialization 3](#_Toc60844605)

[Wat is het? 3](#_Toc60844606)

[Hoe komt her voor in mijn project? 3](#_Toc60844607)

[Hoe voorkom ik het? 3](#_Toc60844608)

[Using Components with Known Vulnerabilities 4](#_Toc60844609)

[Wat is het? 4](#_Toc60844610)

[Hoe komt her voor in mijn project? 4](#_Toc60844611)

[Hoe voorkom ik het? 4](#_Toc60844612)

[Insufficient Logging & Monitoring. 4](#_Toc60844613)

[Wat is het? 4](#_Toc60844614)

[Waarom zou ik loggen % monitoren? 4](#_Toc60844615)

[Hoe komt her voor in mijn project? 4](#_Toc60844616)

# [**Injection**](https://owasp.org/www-project-top-ten/2017/A1_2017-Injection)

## Wat is het?

Injection is dat je een query injecteerd in bijvoorbeeld een input veld. Dus bijvoorbeeld:

; DROP users;

Dan wordt de database users gedropt en kan niemand meer inloggen omdat de user database weg is. Zo kan je data verliezen of nog erger, je data kan gestolen worden als ze de juiste query gebruiken.

## Hoe komt het voor in mijn project?

Omdat ik in mijn project Enitity framework gebruik is het  SQL Injection Proof. Maar als ik zelf custom queries ga schrijven dan moet ik wel rekening houden dat het niet meer SQL injection Proof kan zijn.

## Hoe voorkom ik het?

In c# kan je SQL inejection voorkomen door in de Query parameters te gebruiken. De parameters kun je aanmaken via SqlCommand. Met de code hieronder wordt een query uitgevoerd met parameters in de query.

private static void UpdateDemographics(Int32 customerID,

string demoXml, string connectionString)

{

// Update the demographics for a store, which is stored

// in an xml column.

string commandText = "UPDATE Sales.Store SET Demographics = @demographics "

+ "WHERE CustomerID = @ID;";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(commandText, connection);

command.Parameters.Add("@ID", SqlDbType.Int);

command.Parameters["@ID"].Value = customerID;

// Use AddWithValue to assign Demographics.

// SQL Server will implicitly convert strings into XML.

command.Parameters.AddWithValue("@demographics", demoXml);

try

{

connection.Open();

Int32 rowsAffected = command.ExecuteNonQuery();

Console.WriteLine("RowsAffected: {0}", rowsAffected);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

# [**Broken Authentication**](https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication)

## Wat houd het in

Broken authenitcation houd in dat hackers proberen in te loggen onder een user dat niet van hun is. Dit doen ze op verschillende manieren. Ze proberen alle simpele wachtwoorden uit zoals: wachtwoord, admin en qwerty. Een hacker kan ook al een lijst hebben van de gebruiker. Hij kan dan alle wachtwoorden 1 voor 1 uit gaan testen. Ook kan het voorkomen dat een gebruiker in een openbare ruimte is ingelogd en is vergeten uit te loggen, nu kan de volgende persoon die op de site komt bij de gebruikers zijn account.

## Hoe komt het voor in mijn project?

Omdat ik in mijn project een authenticatie heb kan het bij voor komen dat er een brute force attack gepleegd kan worden. Mijn applicatie gaat veel gebruikt worden op mobiele apparaten dus de kans is klein dat er een persoon ingelogd is die niet ingelogd hoort te zijn.

## Hoe voorkom ik het?

Je voorkomt het niet, maar je kan het hackers wel veel moeilijker maken om bijvoorbeeld 2-step-authenication toe te voegen. Ook kan je de user informeren dat zijn wachtwoord erg zwak is bij het registeren van een account. Je kunt ook rekening houden met de tijd dat de sessie bestaat zodat users die ingelogd zijn in openbare ruimtes en vergeten uit te loggen hun account niet gebruikt kan worden door anderen. Voor brute force attacks te vermilde kan je een delay toevoegen tussen gefaalde inlog pogingen, dan duurt het langer voor een brute force attack om te lukken.

# [**Sensitive Data Exposure**](https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure)

## Wat is het?

Sensitive data exposure is een fout van de ontwikkelaar. Het gaat hier over gevoelige informatie zoals:

* Creditcard informatie
* Gezondheid
* Identiteit

Deze data kan onderschept worden wanneer het van de server naar de gebruiker gestuurd wordt of de database kan gehackt worden en dan de data meteen leesbaar.

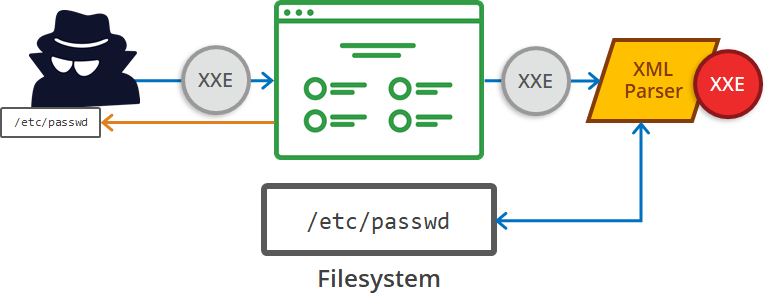
## Hoe voorkom ik het?

Je kunt het voorkomen door de data te encrypten wanneer het van de server naar de gebruiker gaat en/of de data geëncrypt op te slaan.

## Hoe komt het voor in mijn project?

In mijn project heb ik geen gevoelige data opgeslagen.

# [**XML External Entities (XXE)**](https://owasp.org/www-project-top-ten/2017/A4_2017-XML_External_Entities_(XXE))



## Wat is het?

XML External Entities is dat een hacker de verwerking van XML data kan veranderen. Vaak kan de hacker dan de files op de server waar de applicatie staat in zien. Ook kan de hacker bij alle back-ends of losse applicaties waar de applicatie bij kan. Er zijn 4 soorten XXE aanvallen:

* Om files te onvangen:
  + Als de entity de waardes bevat van een file en terug gestuurd wordt in een response.
* SSRF aanval ( server-side request forgery ):
  + Als de entity bestaat uit een URL backend.
* Data out-of-band:
  + De data wordt van de server van de applicatie naar een eigen omgeving gestuurd worden en dan zo uitgelezen kan worden.
* Error Messages:
  + De hacker kan proberen om een error te krijgen en dan zo de data te ontvangen.

## Hoe komt her voor in mijn project?

Het komt niet voor in mijn project omdat ik geen gebruik maar van XML data.

## Hoe voorkom ik het?

De veiligste manier is het uitzetten van DTDs (External Entities), dit kan op de volgende manier. Dit werkt niet bij elke taal het zelfde maar zou er erg op lijken:

factory.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true);

# [**Broken Access Control**](https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control)

## Wat is het?

Broken Access Control is dat gebruikers elkaars data in kunnen zien / aanpassen. Ook kan het voor komen dat ongeautoriseerde gebruikers bepaalde functies kunnen gebruiken waar ze anders meer voor moeten betalen bijvoorbeeld.

## Hoe komt het voor in mijn project?

In mijn project kunnen gebruikers alleen hun eigen bestellingen zien. De gebruiker die een barman is mag van andere gebruikers hun bestellingen aanpassen. Een admin mag ook alle functies gebruiken en mag daarbij ook de producten aanpassen op een aparte website waar alleen de admin in mag loggen.

## Hoe voorkom ik het?

Om Broken Access control te verkomen moet er in de applicatie een check komen of de user is geautoriseerd. Ook moet er gezorgd worden dat alleen de data die de user mag zien verstuurd wordt naar de user.

# [**Security Misconfiguration**](https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration)

## Wat is het?

Security Misconfiguration is het vergeten of het niet goed instellen van de beveiliging in een server. Als een bedrijf denkt dat hij het goed ingesteld heeft maar dat niet zo blijkt te zijn kan het zijn dat je data gestolen kan worden

## Hoe komt het voor in mijn project?

De kans is groot dat dit voorkomt in mijn project, dit is een bekent groot probleem dat moeilijk op te lossen is omdat er veel verschillende frameworks gebruikt worden. De grootste risco ligt dan bij het niet afmaken van configuratie waardoor er verschillende poorten open blijven die niet open hoeven te staan.

## Hoe voorkom ik het?

In ASP.NET kun je het voor komen door:

* Framework
  + Error handling
  + Config Transforms
  + Diagnostics, Logging
  + Web config security
  + Software updates
* Database
  + Acces controls
  + Principle of least privilege
* Front-end
  + Error handling
  + Authentication options

# What Is Cross-Site Scripting? | Cloudflare[**Cross-Site Scripting XSS**](https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS))

## Wat is het?

XSS is een fout in de beveilliging van de webapplicatie. Het probleem kan verookzaakt worden als een Cookie, URL of de URL parameters niet goed verwerkt worden en zo kan dan uiteindelijk de sessiecookies en sessie van een gebruiker over genomen worden. XSS wordt vaak samen gebruikt met Phising. Dit kan dan via een email met een link.

## Hoe komt het voor in mijn project?

Omdat ik voor mijn project Angular gebruik is het heel lastig om XSS toe te passen.

## Hoe voorkom ik het?

In bijvoorbeeld een PHP applicatie kan je URL parameters mee geven:

Localhost:8080?bericht=*<script>alert('XSS');</script>*

1<?php

2

3 **if** ( isset ( $\_GET["BERICHT"] ) )

4 **echo** $\_GET["BERICHT"];

Wanneer je het bericht echo’t op de pagina, dan krijg je een javascript alert die zegt: XSS. Zoals hier boven is te zien. Maar met de code hieronder wordt word er voor gezorgd dat de code niet als een html tag kan worden uitgevoerd maar als plain tekst.

1<?php

2

3 **if** ( isset ( $\_GET["BERICHT"] ) )

4 **echo** htmlentities ( $\_GET["BERICHT"] );

# [**Insecure Deserialization**](https://owasp.org/www-project-top-ten/2017/A8_2017-Insecure_Deserialization)

## OWASP Insecure Deserialization Vulnerability - The CybersploitWat is het?

Insecure Deserialization is dat onbekende of onvertrouwde data gebruikt kan worden voor een DoS acttack (denial of service), om bepaalde code uit te voeren of om de authenticatie te omzeilen. Serialization is het omzetten van een object naar een formaat dat de data doorgestuurd kan worden, zoals JSON, XML of YAML. Deserialization is het omgekeerde van serialization, dan wordt de data omgezet naar een object. Een attacker kan het deserialization proces misbruiken als het niet goed beveiligd is. De attacker kan zo serialized data naar de webapp sturen die dan vervolgens door de gebruiker geserialiseerd wordt en kan de attacker proberen om in de applicatie te komen via de gebruikers zijn computer.,

## Hoe komt het voor in mijn project?

In mijn project wordt er gebruik gemaakt van JSON data. Met in deze JSON staat de data dat nodig is voor in de front-end.

## Hoe voorkom ik het?

Om het te verkomen kun je wanneer je het deserialiseerd zeggen wat voor object het moet worden bijvoorbeeld:

return JsonConvert.DeserializeObject<ProductDTO>(result);

Hier wordt de json gedeserialiseerd naar een ProductDto. Andere data wordt er zo uit gehaald en niet gebruikt

# [**Using Components with Known Vulnerabilities**](https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities)

## Wat is het?

Als je gebruik maakt van libraries of componenten van 3de kan het voorkomen dat er kwetsbaarheden in zitten die hackers kunnen gebruiken om zo je bij je data te komen.

## Hoe komt het voor in mijn project?

## Hoe voorkom ik het?

Je kunt het voorkomen om je libraries op-to-date te houden, zo voorkom je dat hackers je data bij kunnen via backdoors. Je kunt je programma ook scannen met verschillende tools die de vulnerabilities herkennen.

# [**Insufficient Logging & Monitoring**](https://owasp.org/www-project-top-ten/2017/A10_2017-Insufficient_Logging%2526Monitoring).

## Overzicht van Azure Monitor - Azure Monitor | Microsoft DocsWat is het?

Met logging en monitoring houd je alle belangrijke events bij zoals: login, mislukte logins en betalingen. Ook worden alle errors en warnings bijgehouden in de log. Deze logs staan local op de server zodat niemand anders deze logs in kan zien.

## Waarom zou ik loggen % monitoren?

Zorg er voor dat je je applicatie kunt monitoren. Je kunt er in patronen in herkennen. Je kunt er ook bugs uithalen die zorgen dat bepaalde functionaliteiten niet meer goed verwerken.

## Hoe komt het voor in mijn project?

Omdat ik mijn project gehost heb bij Azure heb ik de mogelijkeheid om zo mijn applicatie te monitoren.