|  |
| --- |
| Documentatie WTIS\_  HERKANSING |

Thijs van Alst (1670161)

Docent: Fritz van Deventer

WTIS

25 juni 2025

INHOUDSOPGAVE

[1 Gemaakte keuzes voor de vijf risico’s 4](#_Toc201850924)

[2 SQL injectie 5](#_Toc201850925)

[2.1 Gevolgen van SQL injectie 5](#_Toc201850926)

[Beveiliging in mijn applicatie 5](#_Toc201850927)

[3 Broken authentication 6](#_Toc201850928)

[3.1 Gevolgen broken authentication 6](#_Toc201850929)

[3.2 Beveiliging van mijn applicatie 6](#_Toc201850930)

[4 Sensitive data exposure 7](#_Toc201850931)

[4.1 Gevolgen van sensitive data exposure 7](#_Toc201850932)

[4.2 Beveiliging in mijn applicatie 7](#_Toc201850933)

[5 access control failures 8](#_Toc201850934)

[5.1 Gevolgen van access control failures 8](#_Toc201850935)

[5.2 Beveiliging in mijn applicatie 8](#_Toc201850936)

[6 Cross-site scripting (XSS) 9](#_Toc201850937)

[6.1 Gevolgen van XSS 9](#_Toc201850938)

[6.2 Beveiliging in mijn applicatie 9](#_Toc201850939)

[7 Conclusie 10](#_Toc201850940)

INLEIDING

In dit document worden vijf beveiligingsrisico’s uit de OWASP Top 10 toegelicht en toegepast op de webapplicatie van Pizzeria Sole Machina. Voor elk risico is een risicotabel opgenomen, gevolgd door een uitleg van de gevolgen en een samenvatting van de beveiligingsmaatregelen in de applicatie. Waar mogelijk worden relevante codevoorbeelden uit de gemaakte applicatie genoemd.

# Gemaakte keuzes voor de vijf risico’s

Bij het selecteren van de vijf OWASP-risico’s is gekeken naar de onderdelen van de applicatie die de meeste risico’s met zich meebrengen op het gebied van gebruikersinteractie, gegevensverwerking en rechtenbeheer.

* SQL-injectie is gekozen omdat de applicatie op meerdere plekken invoer verwerkt die door de gebruiker wordt meegestuurd naar de database, zoals bij inloggen en bestellingen plaatsen. Dit is een klassiek risico dat goed te beveiligen is met prepared statements.
* Broken Authentication is gekozen omdat de site een volledig eigen login- en registratiesysteem heeft, inclusief wachtwoordverwerking. Een fout hierin kan leiden tot ongewenste toegang tot gevoelige onderdelen.
* Sensitive Data Exposure is relevant omdat gebruikers persoonlijke gegevens invoeren, zoals hun naam, wachtwoord en afleveradres. Ook al wordt de site niet online gehost, moeten deze gegevens veilig worden verwerkt en opgeslagen.
* Access Control Failures is belangrijk in deze applicatie omdat er onderscheid is tussen klanten, personeel en beheerders. Het is essentieel dat klanten geen toegang krijgen tot beheerfuncties, en dat alleen admins personeel mogen beheren.
* Cross-Site Scripting (XSS) is gekozen omdat de applicatie op meerdere plekken gebruikersdata toont, zoals productnamen, foutmeldingen of bestelgeschiedenis. Zonder ontsmetting kan dit leiden tot het uitvoeren van kwaadaardige scripts.

Door deze vijf risico’s uit te werken, wordt aangetoond dat er goed is nagedacht over de veiligheid van de webapplicatie op plekken waar invoer en verwerking van gebruikersdata centraal staan.

# SQL injectie

Tabel 1 R1: SQL injectie

|  |  |
| --- | --- |
| Risico | R1: SQL-injectie via invoervelden of verborgen POST-data |
| Aanvalstechniek | Broncode-injectie via invoer in bijvoorbeeld login of formuliervelden |
| Kans | Hoog: komt vaak voor zonder prepared statements |
| Gevolg | Hoog, aanvaller kan de database uitlezen, wijzigen of verwijderen |

## Gevolgen van SQL injectie

Zonder goede bescherming tegen SQL injectie kunnen gebruikers met kwaadaardige invoer toegang krijgen tot gegevens van andere gebruikers, inloggen zonder wachtwoord of zelfs volledige tabellen verwijderen of aanpassen.

Beveiliging in mijn applicatie  
Ik gebruik op alle plekken in mijn code prepared statements met placeholders, bijvoorbeeld bij het registreren:

Afbeelding met tekst, schermopname

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 1 Registratie SQL statements

Dit voorkomt dat gebruikers via de invoer velden SQL kunnen manipuleren.

# Broken authentication

Tabel 2 R2: Broken Authentication

|  |  |
| --- | --- |
| Risico | R2: Onveilige loginmechanismen en sessiebeheer |
| Aanvalstechniek | Bruteforce, sessie kapen, geen wachtwoord hashing |
| Kans | Hoog: komt vaak voor bij zelfgebouwde loginmodules |
| Gevolg | Hoog: ongeautoriseerde toegang tot accounts en gegevens |

## Gevolgen broken authentication

Als wachtwoorden niet goed worden opgeslagen of sessies niet goed worden gecontroleerd, kunnen gebruikers in elkaars accounts komen of altijd ingelogd blijven.

## Beveiliging van mijn applicatie

Ik maak gebruik van password\_hash() bij het registreren van gebruiker, daarnaast laat ik hun inloggen met password\_verify(). Waarna sessie gegevens worden opgeslagen met $\_SESSION[“username”] en eventueel gecontroleerd op rol.

Afbeelding met tekst, schermopname, Lettertype

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 2 $\_SESSION code

# Sensitive data exposure

Tabel 3 Sensitive data exposure

|  |  |
| --- | --- |
| Risico | R3: Onvoldoende beveiliging van gevoelige gegevens |
| Aanvalstechniek | Gegevens uitlezen via database, browser of netwerk |
| Kans | Middel: afhankelijk van hosting en opslag |
| Gevolg | Hoog: wachtwoorden of persoonsgegevens kunnen uitlekken |

## Gevolgen van sensitive data exposure

Bij een lek kunnen aanvallers toegang krijgen tot gebruikersnamen, wachtwoorden, adressen en bestelgegevens.

## Beveiliging in mijn applicatie

* Wachtwoorden zijn gehasht met password\_hash()
* Adresgegevens worden alleen tijdelijk bij bestellingen opgeslagen
* In de code worden geen wachtwoorden gelogd of getoond
* Databaseconnectie is gescheiden in db\_connectie.php
* Gebruik maken van unset() statement

Afbeelding met tekst, schermopname, Lettertype

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 3 Unset statement

# access control failures

Tabel 4 R4: Access control failures

|  |  |
| --- | --- |
| Risico | R4: Pagina’s of functies bereikbaar zonder juiste rechten |
| Aanvalstechniek | Directe URL-aanroepen of formulier-manipulatie |
| Kans | Middel: komt vaak voor zonder sessiecontrole |
| Gevolg | Hoog: gebruikers kunnen beheerpagina’s bekijken of aanpassen |

## Gevolgen van access control failures

Bij access control failures kan een gewone klant functionaliteit uitvoeren als admin of personeel. Daarbij dus bestellingen aanpassen of personeel verwijderen.

## Beveiliging in mijn applicatie

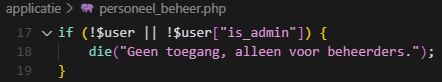
* Bij alle gevoelige pagina’s controleren of de gebruiker is ingelogd en de juiste rol heeft

Afbeelding met schermopname, tekst, Lettertype

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 4 check of er een sessie aanwezig is

* Voor beheerderspagina’s is er een check of is\_admin = 1 is



Figuur 5 controle of iemand toegang heeft

# Cross-site scripting (XSS)

Tabel 5 R5: XSS

|  |  |
| --- | --- |
| Risico | R5: XSS via formulierinvoer of gebruikersgegevens |
| Aanvalstechniek | Invoegen van scripts via formulieren of URL’s |
| Kans | Middel: afhankelijk van waar output niet getrimt wordt |
| Gevolg | Middel tot hoog: sessies kapen, manipulatie van UI |

## Gevolgen van XSS

Als een gebruiker HTML of JavaScript kan invoeren die niet gefilterd wordt, kan dit op andere pagina’s uitgevoerd worden tijdens een redirect. Daardoor kan een andere gebruiker worden misleid of worden overgenomen.

## Beveiliging in mijn applicatie

* Foutmeldingen of bevestigingen worden ook ontsmet weergegeven
* Er worden geen HTML-framenten of user-generated content niet niet ontsmet getoont.
* Alle uitvoer van gebruikersdata gaat via htmlspecialchars();

Afbeelding met schermopname, tekst

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 6 htmlspecialchars()

# Conclusie

Deze vijf beveiligingsrisico’s zijn serieus genomen in het ontwerp en de implementatie van de applicatie. Door gebruik te maken van veilige functies zoals htmlspecialchars(), password\_hash() en prepared statements zijn deze risico’s grotendeels afgehandeld. Andere beveiligingsmaatregelen zoals sessiebeheer, toegangscontrole en duidelijke scheiding tussen rollen zorgen ervoor dat gebruikers veilig gebruik kunnen maken van de webapplicatie.