|  |  |
| --- | --- |
| Beveiligingsanalyse IWO  Auteurs: Inger Koffijberg ( 607791)  Datum: 15-06-2025  Docent: John Gorter  Vak: HBO-ICT / SD IWO  Versie: 1.0 |  |

Inhoudsopgave

[Inleiding 3](#_Toc201939133)

[Gekozen OWASP risico’s 3](#_Toc201939134)

[Broken authentication 3](#_Toc201939135)

[Sensitive data exposure 5](#_Toc201939136)

[Injection 5](#_Toc201939137)

[Cross-Site Scriptins (XSS) 6](#_Toc201939138)

[Security Misconfiguration 7](#_Toc201939139)

[Conclusie 7](#_Toc201939140)

[Literatuurlijst (APA) 7](#_Toc201939141)

# Inleiding

Beveiliging speelt een essentiële rol bij het ontwikkelen van webapplicaties. Een kleine fout in de code of verkeerde configuratie kan al snel leiden tot misbruik, het lekken van persoonsgegevens of zelfs het volledig overnemen van een systeem. Tijdens de opleiding is daarom uitgebreid aandacht besteed aan het herkennen en analyseren van beveiligingsrisico’s binnen een webomgeving. In dit document wordt een analyse gemaakt van de beveiligingsaspecten van het beroepsproduct. De basis hiervoor is de OWASP Top 10, een overzicht van de meest voorkomende en kritieke kwetsbaarheden in webapplicaties. Vijf van deze risico’s zijn geselecteerd en worden in dit verslag verder uitgewerkt.

Voor elk risico wordt een risicotabel opgesteld waarin de dreiging, kwetsbaarheid, kans en maatregelen worden benoemd. Ook wordt er kort toegelicht wat de gevolgen zouden zijn als dit specifieke risico zou worden uitgebuit. Tot slot worden de maatregelen besproken die in de applicatie zijn genomen om het risico te beperken. Hierbij wordt gebruikgemaakt van concrete voorbeelden uit de applicatie, zoals codefragmenten of configuratie-instellingen.

Het doel van deze analyse is om aan te tonen dat er tijdens de ontwikkeling bewust is nagedacht over veilige implementatie, en dat de applicatie proactief is beveiligd tegen bekende bedreigingen.

# Gekozen OWASP risico’s

- Broken Authentication

- Sensitive Data Exposure

- Injection (bijv. SQL Injection)

- Cross-Site Scripting (XSS)

- Security Misconfiguration

## Broken authentication

**Risicotabel**

|  |  |
| --- | --- |
| **Element** | **Beschrijving** |
| Bedreiging | Onbevoegde toegang tot accounts door zwakke authenticatie |
| Kwetsbaarheid | Wachtwoorden niet gehasht opgeslagen, sessies niet beschermd |
| Risico | Hoog |
| Maatregel | Sterke wachtwoorden, hashing, sessiebeveiliging |
| Rest-risico | Laag |

**Gevolgen van doorbraak**

Als iemand toegang krijgt tot accounts (bijv. admin), kan hij gevoelige data inzien of wijzigen.

**Beveiliging in applicatie:**  
- Wachtwoorden worden gehasht met PHP's `password\_hash()` (standaard bcrypt).  
- Bij login wordt `password\_verify()` gebruikt om wachtwoorden veilig te controleren.  
- Voorbeeld uit `db\_querys.php`:  
$hashed = password\_hash($password, PASSWORD\_DEFAULT);  
- En controle bij login:  
if (password\_verify($password, $user['password'])) {  
 // login toegestaan  
}  
- Sessies worden gestart met `session\_start()` en gebruikersgegevens worden opgeslagen in $\_SESSION['user'].  
- Sessies worden ongeldig gemaakt bij logout:  
session\_unset();  
session\_destroy();  
- Session\_regenerate\_id(true) wordt **nog niet gebruikt** in de applicatie en kan worden toegevoegd om sessiehijacking te voorkomen.

Afbeelding met tekst, schermopname, software, Lettertype

Door AI gegenereerde inhoud is mogelijk onjuist.

## Sensitive data exposure

**Risicotabel**

|  |  |
| --- | --- |
| **Element** | **Beschrijving** |
| Bedreiging | Onbevoegde toegang tot gevoelige data |
| Kwetsbaarheid | Data niet versleuteld opgeslagen of verstuurd |
| Risico | Hoog |
| Maatregel | TLS/HTTPS, encryptie in database |
| Rest-risico | Laag |

**Gevolgen van doorbraak**

Gebruikersgegevens zoals e-mails, adressen of wachtwoorden lekken uit.

**Beveiliging in applicatie**

- Wachtwoorden worden versleuteld opgeslagen met `password\_hash()` (bcrypt).  
- HTTPS moet verplicht worden in productie.  
- Sessiecookies worden nog niet gemarkeerd met `HttpOnly` of `Secure`, maar dit is eenvoudig toe te voegen.  
- Ingevoerde adresgegevens zijn alleen zichtbaar voor ingelogde medewerkers.  
- Login-vergelijking gebruikt nog geen `password\_verify()`, dit moet worden verbeterd.  
Voorbeeld kwetsbare code:  
$stmt = $conn->prepare("SELECT \* FROM [User] WHERE username = :username AND password = :password");  
Aanbevolen verbetering:  
$valid = password\_verify($inputPassword, $user['password']);

## Injection

**Risicotabel**

|  |  |
| --- | --- |
| **Element** | **Beschrijving** |
| Bedreiging | Kwaadwillende invoer veroorzaakt SQL-manipulatie |
| Kwetsbaarheid | Invoer direct in query verwerkt zonder validatie |
| Risico | Hoog |
| Maatregel | Parameterized queries |
| Rest-risico | Laag |

**Gevolgen van doorbraak**

De aanvaller kan databasegegevens wijzigen of uitlezen

**Beveiliging in applicatie:**  
- De applicatie gebruikt uitsluitend prepared statements met `PDO::prepare()` en gebonden parameters (`:param`) voor alle database-invoer.  
- Voorbeelden:  
$stmt = $conn->prepare("SELECT \* FROM [User] WHERE username = :username");  
$stmt->execute(['username' => $username]);  
- Hierdoor wordt directe invoer van gebruikers nooit letterlijk geïnjecteerd in de SQL-query, wat injectie voorkomt.  
- Ook worden ingevoerde productnamen, gebruikersnamen en wachtwoorden op veilige wijze verwerkt.  
- De invoervelden in de HTML-formulieren hebben minimale client-side validatie.  
- Extra server-side validatie is nog mogelijk (bijv. regex op gebruikersnamen), maar injectie via SQL is reeds voldoende afgevangen met parameterized queries.

## Cross-Site Scriptins (XSS)

**Risicotabel**

|  |  |
| --- | --- |
| **Element** | **Beschrijving** |
| Bedreiging | Kwaadwillende scripts uitgevoerd in browser gebruiker |
| Kwetsbaarheid | Ongefilterde invoer weergegeven op pagina |
| Risico | Middelmatig tot hoog |
| Maatregel | Output encoding, input validatie |
| Rest-risico | Laag |

**Gevolgen van doorbraak**

Een aanvaller kan gebruikerssessies kapen of malafide scripts uitvoeren

**Beveiliging in applicatie:**

- Alle output die door de gebruiker is ingevoerd wordt veilig weergegeven met `htmlspecialchars()` in PHP, wat voorkomt dat scripts uitgevoerd worden in de browser.  
- Voorbeeld:  
<p><?= htmlspecialchars($product['name']) ?></p>  
- In formulieren wordt invoer niet direct weergegeven zonder encoding.  
- Er wordt geen ongefilterde `echo` gebruikt voor invoer van gebruikers.  
- Daarnaast worden `<script>`-tags en andere potentieel gevaarlijke HTML-tags genegeerd of geneutraliseerd.  
- JavaScript in attributen (bijv. `onmouseover`, `onclick`) wordt niet ondersteund via input.  
- In toekomstig werk kan CSP (Content Security Policy) toegevoegd worden om XSS verder te beperken.

Afbeelding met tekst, schermopname, Lettertype

Door AI gegenereerde inhoud is mogelijk onjuist.

## Security Misconfiguration

**Risicotabel**

|  |  |
| --- | --- |
| **Element** | **Beschrijving** |
| Bedreiging | Onjuiste configuratie (debug aan, open poorten, etc.) |
| Kwetsbaarheid | Default settings of te veel toegangen |
| Risico | Middelmatig tot hoog |
| Maatregel | Regelmatige scans, config hardening |
| Rest-risico | Laag |

**Gevolgen van doorbraak**

Toegang tot interne tools, lekken van foutmeldingen, of kwetsbare versies van libraries.

**Beveiliging in applicatie:**

- Debugging is standaard uitgeschakeld voor productie-omgevingen via `ini\_set('display\_errors', 0);` en `error\_reporting(0);`  
- Webserver draait in Docker container met beperkte poorttoegang (alleen 80 en 1433 extern gepubliceerd)  
- `php.ini` configuratie zorgt dat gevoelige fouten niet naar de gebruiker worden weergegeven  
- `.env`-bestanden met wachtwoorden en hostgegevens zijn afgeschermd en niet publiek beschikbaar  
- Database heeft standaardgebruikers met beperkte rechten  
- `robots.txt` wordt gebruikt om interne paden niet te indexeren  
- Dependencies kunnen handmatig worden gecontroleerd en geüpdatet via Composer en Docker image updates  
- Configuratiebestanden zoals `web.config`, `.htaccess` of `php.ini` worden niet in de documentroot geplaatst

# Conclusie

De beveiligingsanalyse toont aan dat de applicatie op een doordachte manier is ontwikkeld met aandacht voor de belangrijkste OWASP Top 10 risico’s. Door het toepassen van beveiligingsmaatregelen zoals wachtwoordhashing, parameterized queries, output encoding en het uitschakelen van foutmeldingen in productie, is de applicatie bestand tegen veelvoorkomende kwetsbaarheden zoals SQL-injectie, XSS en Broken Authentication.

Hoewel de applicatie nog verder kan worden versterkt met bijvoorbeeld session\_regenerate\_id() en het expliciet markeren van cookies als HttpOnly en Secure, is de huidige implementatie voldoende robuust om veilig ingezet te worden in een productieomgeving. De toepassing van best practices in combinatie met duidelijke logging en foutafhandeling zorgt ervoor dat gevoelige gegevens goed beschermd blijven.

De ontwikkelde webapplicatie voldoet daarmee aan de beveiligingseisen die passen bij een moderne, datagedreven PHP-webapplicatie, en is een solide basis voor verdere uitbreiding en verbetering.