|  |
| --- |
| WTIS DOCUMENTATIERAPPORT \_  Versie 1 |

**Naam:** Sasha van Ree

**Studentnummer:** 2111744

**Docent:** Fritz van Deventer

Datum: 20-05-24

**Gelegenheid 3**

INHOUDSOPGAVE

[INLEIDING 3](#_Toc201951344)

[1 Vijf risico’s 4](#_Toc201951345)

[1.1 Broken Accesss Control 4](#_Toc201951346)

[1.1.1 Mijn maatregelen 4](#_Toc201951347)

[1.2 Cryptographic Failures 5](#_Toc201951348)

[1.2.1 Mijn maatregelen 5](#_Toc201951349)

[1.3 Injection (SQL) 6](#_Toc201951350)

[1.3.1 Mijn Maatregelen 6](#_Toc201951351)

[1.4 Insecure Design 7](#_Toc201951352)

[1.4.1 Mijn maatregelen 7](#_Toc201951353)

[1.5 Identification and Authentication Failures 8](#_Toc201951354)

[1.5.1 Mijn maatregelen 8](#_Toc201951355)

[2 Bijlagen 9](#_Toc201951356)

[2.1 Broken Access Control 9](#_Toc201951357)

[2.2 Cryptographic Failures 10](#_Toc201951358)

[2.3 Injection (SQL) 11](#_Toc201951359)

[2.4 Insecure Design 12](#_Toc201951360)

[2.5 Identification and Authentication Failures 13](#_Toc201951361)

[3 zelf ingevuld beoordelingsmodel 14](#_Toc201951362)

[4 Bibliografie 15](#_Toc201951363)

[(OWASP, 2021) 16](#_Toc201951364)

INLEIDING

Voor de herkansing van WTIS heb ik de casus van Pizzaria Sole Machina uitgewerkt. Hiervoor moest een website gebouwd worden waarin bestellingen geplaatst en ingezien kunnen worden voor zowel de klant als het personeel.

In dit rapport bespreek ik vijf risico’s uit de top 10 van OWASP, Open Worldwide Application Security Project, door middel van een risicotabel. Daarna leg ik uit hoe ik binnen mijn eigen applicatie deze vijf risico’s heb opgevangen door voorbeelden te geven van mijn code met een korte uitleg.

Vorig jaar heb ik WTUX al gehaald dus ik hoef niet beoordeeld te worden op de html en CSS code binnen mijn applicatie.

# Vijf risico’s

Per risico laat ik stukken code zien waarmee ik de volgende vijf risico’s heb opgelost. Deze staan in hoofdstuk 2 “Bijlagen”.

## Broken Accesss Control

|  |  |
| --- | --- |
| Risico | Aanvallers kunnen makkelijker zonder autorisatie toegang krijgen tot gevoelige informatie. |
| Aanvalstechniek | * Aanvallers kunnen sessies van andere gebruikers gebruiken om zo toegang te krijgen tot functies en gegevens. * Aanvallers kunnen objecten of gegevens direct manipuleren via URL of andere middelen. * Aanvallers kunnen een beheerderstaak uitvoeren zonder autorisatie door een script te injecteren. |
| Kans | Hoog: Staat op nummer 1 binnen de OWASP top 10 meest voorkomende veiligheidsrisico’s. |
| Gevolg | Datalekken kunnen plaatsvinden, gegevens kunnen gewijzigd of verwijderd worden, ongeautoriseerdeacties kunnen uitgevoerd worden wat uiteindelijk kan leiden tot een complete overname van de applicatie of het systeem. |

### Mijn maatregelen

*Zie figuur 1, 2 en 3 in hoofdstuk 2 Bijlagen.*

Door te voorkomen dat aanvallers ongeautoriseerd in kunnen loggen als personeel heb ik op de login.php redirects geïmplementeerd op basis van de rol van de gebruiker.

In functions.php heb ik een functie loginUser geïmplementeerd die bij aanroep de username en rol van de gebruiker ophaalt en deze toevoegt aan de $\_SESSION array. Ik roep loginUser vervolgens aan op de login.php pagina waarop ik, op basis van de rol van de gebruiker, bepaal welke pagina de gebruiker terecht komt als de gegevens correct zijn ingevoerd.

Daarnaast heb ik ervoor gezorgd dat je via de URL niet naar personeel pagina’s kan navigeren, maar in plaats daarvan naar een unauthorized.php pagina wordt gestuurd die weer een link naar de login.php bevat. Op deze manier kunnen gebruikers niet zomaar ongeautoriseerd op pagina’s komen.

## Cryptographic Failures

|  |  |
| --- | --- |
| Risico | Aanvallers kunnen makkelijker gevoelige informatie stelen, misbruiken of manipuleren. |
| Aanvalstechniek | Hangt af van de kwetsbaarheid.   * Aanvallers kunnen verschillende tools gebruiken voor het breken van slechte encryptie algoritmes, zoals MD5, SHA-1 of RC4 * Aanvaller kan netwerkverkeer via HTTP (in plaats van HTTPS) onderscheppen en zo gevoelige gegevens, zoals wachtwoorden, sessietokens of creditcardgegevens inzien en manipuleren. |
| Kans | Hoog: Staat op nummer 2 binnen de OWASP top 10 meest voorkomende veiligheidsrisico’s. |
| Gevolg | Gegevens kunnen blootgesteld worden door datalekken. Aanvallers kunnen identificatiegegevens stelen en zo toegang krijgen tot meerdere gebruiksaccounts. Bedrijven kunnen financiële verliezen leiden door frauduleuze transacties of het lekken van financiële informatie. |

### Mijn maatregelen

*Zie figuur 4 en 5 in hoofdstuk 2 Bijlagen.*

Binnen mijn applicatie heb ik ervoor gekozen dat gebruikers bij registratie een uniek sterk wachtwoord moeten bedenken, zodat wachtwoorden minder makkelijk gekraakt kunnen worden. Het wachtwoord moet ten minste één kleine letter, één hoofdletter, één cijfer, één speciaal teken en een minimale lengte van 8 tekens hebben.

Vervolgens worden deze sterke wachtwoorden door middel van password\_hash() opgeslagen in de database. De wachtwoorden zijn op deze manier een stuk lastiger te kraken en zijn in het geval van een datalek niet meteen blootgesteld.

## Injection (SQL)

|  |  |
| --- | --- |
| Risico | Aanvallers kunnen ongeautoriseerde toegang krijgen tot de database. |
| Aanvalstechniek | De aanvaller gebruikt kwaadaardige code om de database te kunnen manipuleren en zo toegang te krijgen tot gevoelige informatie binnen de database. |
| Kans | Hoog: Staat op nummer 3 binnen de OWASP top 10 meest voorkomende veiligheidsrisico’s. |
| Gevolg | Kan leiden tot datalekken, datacorruptie en volledig verlies over de controle van het systeem. |

### Mijn Maatregelen

*Zie figuur 6 en 7 in hoofdstuk 2 Bijlagen.*

Om te voorkomen dat aanvallers door middel van injectie toegang kunnen krijgen tot de database heb ik prepared statements gebruikt binnen mijn applicatie.

De query’s worden vooraf gecompileerd door de database, waarbij ik placeholders (:naam) gebruik in plaats van de directe waardes. De data wordt later gebonden aan de placeholders wat ervoor zorgt dat de scheiding tussen de data laag en applicatie laag niet vervaagd.

Omdat de gebruikersinvoer nooit direct in de query komt, kan geen SQL-code geïnjecteerd worden.

## Insecure Design

|  |  |
| --- | --- |
| Risico | Aanvallers kunnen acties uitvoeren waarvoor ze geen recht hebben en zo toegang krijgen tot gevoelige informatie. |
| Aanvalstechniek | Aanvallers focussen zich op logische zwakheden in de werking van de applicatie.   * Geen limieten op inlogpogingen wat brute-force aanvallen mogelijk maakt voor de aanvaller. * Aanvaller kan admin-functionaliteit activeren door URL-manipulatie. |
| Kans | Hoog: Staat op nummer 4 binnen de OWASP top 10 meest voorkomende veiligheidsrisico’s. |
| Gevolg | Datalekken, Overbelasting van het systeem, accounts worden gekraakt. De aanvaller kan uiteindelijk complete controle krijgen over de applicatie en gegevens verwijderen en aanpassen. |

### Mijn maatregelen

*Zie figuur 8 en 9 in hoofdstuk 2 Bijlagen.*

Binnen mijn applicatie heb ik er rekening mee gehouden dat op elke belangrijke actie die gedaan kan worden binnen de applicatie gecheckt wordt op autorisatie. Pagina’s kunnen zonder een geldige inlog niet zomaar bereikbaar zijn via de URL. Al eerder uitgelegd maak ik gebruik van gehashte wachtwoorden die vergeleken kunnen worden met password\_verify() en gebruik ik htmlspecialchars bij het weergeven van gebruikersinvoer.

## Identification and Authentication Failures

|  |  |
| --- | --- |
| Risico | Aanvallers zijn in staat wachtwoorden, keys of session tokens en andere gebruikersaccount informatie te achterhalen om zo ongeautoriseerde toegang te krijgen tot data en het systeem. |
| Aanvalstechniek | * Aanvallers kunnen makkelijk zwakke wachtwoorden, zoals 123456 achterhalen. * Aanvallers kunnen met brute-force aanvallen alle mogelijke combinaties van karakters proberen om een wachtwoord te achterhalen. |
| Kans | Middel: Staat op nummer 4 binnen de OWASP top 10 meest voorkomende veiligheidsrisico’s. |
| Gevolg | Aanvallers kunnen de rol van een geautoriseerde gebruiker op hen nemen wat kan leiden tot ongeautoriseerde toegang tot gevoelige gegevens en systemen. |

### Mijn maatregelen

*Zie figuur 10 in hoofdstuk 2 Bijlagen.*

Om mijn pagina’s te beveiligen tegen identification en authentication failures check ik op elke desbetreffende pagina of de gebruiker ingelogd is of niet. Zo zorg ik ervoor dat pagina’s niet zomaar bereikbaar zijn zonder een geldige inlog.

# Bijlagen

## Broken Access Control

Afbeelding met tekst, schermopname, software, Multimediasoftware

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 1 loginUser in functions.php

Afbeelding met tekst, schermopname

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 2 Aanroep loginUser in login.php

Afbeelding met schermopname

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 3 Redirect naar unauthorized.php

## Cryptographic Failures

Afbeelding met tekst, schermopname, software, Lettertype

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 4 registerUser in functions.php

Afbeelding met tekst, schermopname, software

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 5 wachtwoord controle in register.php

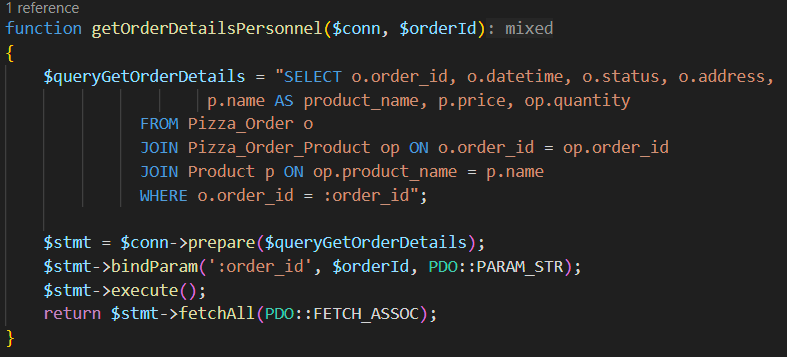
## Injection (SQL)

Voorbeelden van placeholders en prepared statements.

Afbeelding met tekst, schermopname, software

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 6 placeOrder in functions.php



Figuur 7 getOrderDetailsPersonnel in functions.php

## Insecure Design

Afbeelding met tekst, schermopname, Lettertype

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 8 htmlspecialchars gebruik in register.php

Afbeelding met tekst, schermopname, software, Multimediasoftware

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 9 gebruik van password\_verify in loginUser in functions.php

## Identification and Authentication Failures

Afbeelding met tekst, schermopname, Lettertype

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 10 login check op elke desbetreffende pagina

# zelf ingevuld beoordelingsmodel

**Knock-out-criteria**

* Controlekaart: Voldoende
* Risicoanalyse: Voldoende
* Juist gebruik van HTTP: Voldoende

**Multi-tier**

* Multi-tier architectuur: Voldoende

**Security**

* Doorvoeren maatregelen: Voldoende

**Applicatie voldoet aan specificatie**

* Conform specificaties: Voldoende

**Software kwaliteit**

* Herbruikbare onderdelen: Voldoende
* Gestructureerde opzet: Voldoende
* Huidige standaarden: Voldoende

**Databasegebruik**

* Werkt met database: Voldoende
* Efficiënte query’s: Voldoende

# Bibliografie

OWASP. (2021). *https://www.owasptopten.org*. Opgehaald van OWASP Top 10: https://www.owasptopten.org/the-release-of-the-owasp-top-10-2021

(OWASP, 2021)

