|  |
| --- |
| Gelreairport\_  Rapport over beveiligingsrisico’s en zelfbeoordeling |

Vera Wandel

12 januari 2024

INHOUDSOPGAVE

[1 Risicotabellen 4](#_Toc155987109)

[1.1 Broken Access Control 4](#_Toc155987110)

[1.2 Injection 4](#_Toc155987111)

[1.3 Insecure design 5](#_Toc155987112)

[1.4 Security Misconfiguration 6](#_Toc155987113)

[1.5 Identification and Authentication Failures 6](#_Toc155987114)

[2 Zelfbeoordeling 7](#_Toc155987115)

[3 Literatuurlijst 8](#_Toc155987116)

INLEIDING

In dit rapport staan de risico tabellen van vijf van de tien risico’s uit de OWASP top 10. Bij deze tabellen staan ook de gevolgen beschreven en hoe de risico’s kunnen worden voorkomen. Als laatste staat er ook bij hoe de oplossingen zijn verwerkt in mijn eigen applicatie. In het laatste hoofdstuk staat een zelfbeoordeling waarin er per criterium wordt aangegeven wat het verwachte cijfer is.

# Risicotabellen

Tabel 1: Broken Access Control

## Broken Access Control

|  |  |
| --- | --- |
| Risico | R1: Mensen kunnen bij gevoelige data waar ze niet bij horen te komen. |
| Aanvalstechniek | Admin control is slecht beveiligd |
| Kans | **Hoog:** Staat op nummer één van de OWASP top 10(A01: 2021). |
| Gevolg | **Hoog:** Mensen kunnen misbruik maken van het systeem en de data manipuleren. |

Tabel 1: Broken Acces Control

Het gevolg van een doorbraak bij dit risico is dat aanvallers misbruik kunnen maken van het systeem en de data kunnen manipuleren. Dit kan leiden tot datalekken en het stelen van persoonsgegevens.

Om dit risico te voorkomen is het belangrijk dat de gebruikersinvoer wordt gecontroleerd. Ook moet de toegang van een bepaalde pagina of functie worden geblokkeerd, tot dat de gebruiker is geverifieerd. In mijn applicatie controleer ik de gegevensinvoer van formulieren door if statements toe te voegen die controleren of de gebruiker geen verkeerde data invoert.

A screen shot of a computer code

Description automatically generated

Figuur 1: If statements die de gebruikers invoer controleren.

## Injection

|  |  |
| --- | --- |
| Risico | R1: SQL-injectie vanuit views met logincomponent. |
| Aanvalstechniek | Broncode injectie: SQL. |
| Kans | **Hoog:** Staat op nummer drie van de OWASP top 10(A01: 2021). |
| Gevolg | **Hoog:** PHP-broncode die het SQL statement uitvoert heeft minstens de rechten om de hele gebruikerstabel uit te lezen. |

Tabel 2: Injection

Het gevolg van SQL Injection is dat de aanvaller toegang kan krijgen tot gevoelige informatie, zoals gebruikers tabellen en kan deze ook openbaar maken. Ook kunnen aanvallers beheerders worden van het systeem en afwijzingsproblemen veroorzaken.

Een manier om SQL Injection te voorkomen is door de gebruikersinvoer te valideren door middel van prepared statements. Wanneer deze worden gebruikt dan zal de database verschil uit maken tussen code en gegevens, ongeacht wat de gebruiker invoert. Ze zorgen ervoor dat de gebruiker de query’s niet kan wijzigen. Zo ziet een prepared statement er uit:  
A computer screen shot of a computer

Description automatically generated

Figuur 2: Prepared statement

## Insecure design

|  |  |
| --- | --- |
| Risico | R1: Ongeautoriseerde toegang, gegevens lekken en reputatieschade. |
| Aanvalstechniek | URL bewerken, geautomatiseerde scripts en injectie. |
| Kans | **Hoog:** Staat op nummer vier van de OWASP top 10(A01: 2021). |
| Gevolg | Toegang tot sensitieve informatie en manipuleren van de data. |

Tabel 3: Insecure design

Het gevolg van insecure design kan zijn dat aanvallers alle kaartjes van bedrijven op ‘kopen’ wat er voor zorgt dat het bedrijf veel verlies lijdt. Ze kunnen er dus ook voor zorgen dat bepaalde producten erg schaars worden waardoor kopers ze voor een veel hogere prijs moeten inkopen.

Om dit risico te voorkomen is het handig om in de code verschillende lagen aan te houden waardoor de ontwikkelaar het overzicht behoudt. In mijn code is dat gedaan door de applicatie laag, datalaag en presentatie laag gescheiden te houden. De html code staat altijd onder de PHP code.

## Security Misconfiguration

|  |  |
| --- | --- |
| Risico | R1: Goed versterkende security verspreid over de hele applicatie ontbreekt. |
| Aanvalstechniek | Beveiligingsmaatregelen zijn niet optimaal beveiligd bijvoorbeeld door het gebruik van een standaard setting, waardoor het voor aanvallers makkelijker is om in een applicatie te komen. |
| Kans | **Gemiddeld tot hoog:** Staat op nummer vijf van de OWASP top 10(A01: 2021). |
| Gevolg | **Hoog:** Aanvallers kunnen in de applicatie komen. |

Tabel 4: Security Misconfiguration

Een doorbraak bij dit risico kan zorgen voor gegevensverlies of diefstal van gevoelige gegevens. Dit kan weer leiden tot financiële verliezen door bijvoorbeeld boetes en reputatie schade.

Om de kans op Security Misconfiguration klein te houden is het noodzakelijk om onnodige poorten gesloten te houden, het onmogelijk maken om onnodige services uit te voeren, over gebleven pagina’s niet toegankelijk maken en ongebruikte accounts met bepaalde rechten te verwijderen. In mijn programma is het alleen mogelijk om bepaalde services te gebruiken, zoals het registreren van een nieuwe passagier of medewerker, als het formulier is ingevuld. Dit is gedaan door middel van een if statement. Pas als de gebruiker op verzenden klikt wordt de service uitgevoerd.



Figuur 3: If statement waarbij de code pas wordt uitgevoerd als de gebruiker op verzenden klikt.

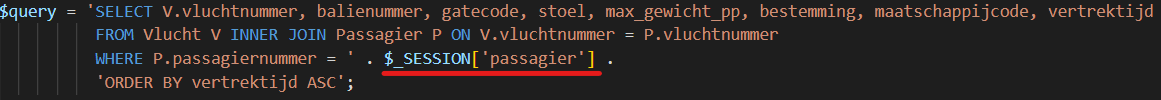
## Identification and Authentication Failures

|  |  |
| --- | --- |
| Risico | R1: Gebrek aan authenticatiemethodes. |
| Aanvalstechniek | Brute force en session hijacking. |
| Kans | **Gemiddeld:** Staat op nummer zeven van de OWASP top 10(A01: 2021). |
| Gevolg | **Hoog:** Datalek, identiteitsfraude. |

Tabel 5: Identification and Authentication Failures

Een doorbraak van dit risico zorgt ervoor dat een aanvaller de identiteit van een andere gebruiker kan gebruiken om zo data te stelen of het systeem op andere manieren in gevaar te brengen.

Om dit risico te voorkomen is het belangrijk om de gebruiker sessies te verifiëren, zodat een aanvaller niet het vorige sessie-id kan gebruiken om zo toegang te krijgen tot sensitieve informatie. In mijn programma is dat gedaan door bij elke pagina met sensitieve data en de weergave van deze data het sessie id te checken. Hierdoor wordt er automatisch geverifiërend of de gebruiker een passagier of een medewerker is en tot welke pagina’s hij wel of geen toegang heeft.



Figuur 4: In de query wordt gecheckt of het passagiernummer wel hetzelfde is als het passagiernummer van de sessie.

# Zelfbeoordeling

Bij het maken van dit project ben ik tegen veel problemen aangelopen en ik ben dan ook niet tevreden met het eindresultaat. Er zijn namelijk bepaalde functies die niet werken naar behoren, zoals het registreren en inloggen. Hierdoor werkt ook de controle met sessie id niet.

|  |  |
| --- | --- |
| Criterium | Cijfer |
| Controle kaart documenten | Voldoende |
| Risico analyse | Voldoende |
| Juist gebruik van HTTP | Voldoende |
| Multi-tier architectuur | 6 |
| Doorvoeren maatregelen | 8 |
| Conform specificaties | 8 |
| Herbruikbare onderdelen |  |
| Gestructureerde opzet | 6 |
| Huidige standaarden | 6 |
| Werkt met database | 6 |
| Efficiënte queries | 6 |

# Literatuurlijst

* *SQL SELECT TOP, LIMIT, FETCH FIRST ROWS ONLY, ROWNUM*. (z.d.). Geraadpleegd op 5 januari 2024, van https://www.w3schools.com/sql/sql\_top.asp
* *SQL Server CURRENT\_TIMESTAMP Function*. (z.d.). Geraadpleegd op 7 januari 2024, van https://www.w3schools.com/sql/func\_sqlserver\_current\_timestamp.asp
* *HTML5 Time element in form with milliseconds*. (z.d.). Stack Overflow. Geraadpleegd op 7 januari 2024, van <https://stackoverflow.com/questions/21127366/html5-time-element-in-form-with-milliseconds>
* *SQL UNION Operator*. (z.d.). Geraadpleegd op 11 januari 2024, van https://www.w3schools.com/sql/sql\_union.asp
* *SQL Injection | OWASP Foundation*. (z.d.). Geraadpleegd op 12 januari 2024, van https://owasp.org/www-community/attacks/SQL\_Injection#:~:text=SQL%20injection%20attacks%20allow%20attackers,administrators%20of%20the%20database%20server.
* Popa, M. (2023, 8 december). *What is broken access control and how to keep your organization safe?* Heimdal Security Blog. Geraadpleegd op 12 januari 2024, van https://heimdalsecurity.com/blog/what-is-broken-access-control/#:~:text=Broken%20access%20control%20can%20lead,or%20data%20they%20have%20accessed.
* *A04 Insecure Design - OWASP Top 10:2021*. (z.d.). Geraadpleegd op 12 januari 2024, van https://owasp.org/Top10/A04\_2021-Insecure\_Design/
* *M8: Security Misconfiguration | OWASP Foundation*. (z.d.). Geraadpleegd op 12 januari 2024, van https://owasp.org/www-project-mobile-top-10/2023-risks/m8-security-misconfiguration#:~:text=Financial%20loss%3A%20Breaches%20resulting%20from,to%20legal%20and%20financial%20consequences.
* *What is broken access control vulnerability and how to prevent it - Authgear*. (z.d.). Geraadpleegd op 12 januari 2024, van https://www.authgear.com/post/what-is-broken-access-control-vulnerability-and-how-to-prevent-it#:~:text=The%20following%20are%20some%20steps,in%20can%20prevent%20the%20attack.
* *SQL Injection Prevention - OWASP Cheat Sheet Series*. (z.d.). Geraadpleegd op 12 januari 2024, van https://cheatsheetseries.owasp.org/cheatsheets/SQL\_Injection\_Prevention\_Cheat\_Sheet.html
* *A04 Insecure Design - OWASP Top 10:2021*. (z.d.-b). Geraadpleegd op 12 januari 2024, van https://owasp.org/Top10/A04\_2021-Insecure\_Design/#how-to-prevent
* Vojtko, M. (2020, 28 december). *How to prevent security misconfiguration*. Hashed Out by The SSL StoreTM. Geraadpleegd op 12 januari 2024, van <https://www.thesslstore.com/blog/how-to-prevent-security-misconfiguration/>
* Hanaim-Webtech. (z.d.). *GitHub - hanaim-webtech/WT-is-beroepsproduct-ENV: WT-Is environment for Dev*. GitHub. Geraadpleegd op 6 januari 2024, van https://github.com/hanaim-webtech/wt-is-beroepsproduct-env
* HAN. (z.d.). *Beroepsproduct - Web Tech: Implementatie & Security (Main)*. Geraadpleegd op 6 januari 2024, van <https://hanaim-webtech.github.io/webtech/wtis/beroepsproduct.html>

