|  |
| --- |
| Documentatie Beveiliging\_  WTIS |

Tren Verheijen (2113801)

24 januari 2025

INHOUDSOPGAVE

[1 analyseren risico’s 3](#_Toc124333034)

# analyseren risico’s

|  |  |
| --- | --- |
| Risico | Broken access control |
| Aanvalstechniek | Misbruikt macht of toegang die een gebruiker niet hoort te hebben. |
| Kans | Hoog: nummer 1 van de OWASP 10. |
| Gevolg | Hoog: Een aanvaller kan gevoelige informatie bekijken of manipuleren en zelfs volledige databases verwijderen.  . |

• Ik gebruik een toegangscontrolemechanisme waarbij alleen ingelogde gebruikers met de juiste rol toegang krijgen tot bepaalde acties.

• Voorbeeld:

function controleerToegang($verwachteRol = null) {

if (session\_status() === PHP\_SESSION\_NONE) {

session\_start();

}

if (!isset($\_SESSION['username']) || !isset($\_SESSION['role'])) {

die("

U moet ingelogd zijn om deze pagina te bekijken.

<br><br>

<a href='../presentatie/klantLogin.php'>Klik hier om in te loggen</a>.

");

}

if ($verwachteRol !== null && $\_SESSION['role'] !== $verwachteRol) {

die("

U heeft geen toegang tot deze pagina.

<br><br>

<a href='../presentatie/menu.php'>Terug naar het menu</a>.

");

}

}

|  |  |
| --- | --- |
| Risico | Cryptographic failure |
| Aanvalstechniek | Achterhalen van inloggegevens door zwakke encryptie. |
| Kans | Hoog: nummer 2 van de OWASP 10. |
| Gevolg | Hoog: – Als de encryptie wordt ondermijnd, kan een aanvaller volledige toegang krijgen tot accounts met hoge privileges. |

* Alle wachtwoorden worden gehashed met een veilig algoritme voordat ze in de database worden opgeslagen. Dit maakt het vrijwel onmogelijk om de originele wachtwoorden te achterhalen, zelfs als de database wordt gelekt.
* Voorbeeld:

$hashedPassword = password\_hash($password, PASSWORD\_DEFAULT);

$stmt = $db->prepare("INSERT INTO [User] (username, password, first\_name, last\_name, address, role) VALUES (?, ?, ?, ?, ?, ?)");

$stmt->execute([$username, $hashedPassword, $firstName, $lastName, $address, $role]);

|  |  |
| --- | --- |
| Risico | SQL-injectie |
| Aanvalstechniek | Broncodeinjectie via SQL |
| Kans | Hoog: nummer 3 van de OWASP 10. |
| Gevolg | Hoog: Een aanvaller kan SQL-query’s manipuleren om volledige tabellen of gevoelige gegevens zoals gebruikersinformatie uit te lezen. |

In mijn project maak ik gebruik van prepared statements en geparameteriseerde queries om ervoor te zorgen dat gebruikersinput nooit rechtstreeks in een SQL-query wordt geplaatst. Hierdoor is het niet mogelijk om SQL-injecties uit te voeren.

• Bijvoorbeeld in de functie haalGebruikerOp gebruik ik de prepare en execute methoden van PDO:

$stmt = $db->prepare("SELECT username, password, role FROM [User] WHERE username = ? AND role = ?");

$stmt->execute([$username, $role]);

|  |  |
| --- | --- |
| Risico | Insecure design |
| Aanvalstechniek | Gebruik maken van informatie die een website vrijgeeft, maar niet zou moeten. |
| Kans | Hoog: nummer 4 van de OWASP 10. |
| Gevolg | Middel: Een aanvaller kan inzicht krijgen in de architectuur van de website en aanvalspunten ontdekken. |

• In plaats van interne foutmeldingen direct weer te geven, stuur ik duidelijke maar niet-specifieke foutmeldingen naar de gebruiker. Hiermee voorkom ik dat gevoelige technische details worden gelekt.

• Voorbeeld:

try {

// Code om een actie uit te voeren

} catch (Exception $e) {

die("Er is een fout opgetreden. Probeer het later opnieuw.");

}

|  |  |
| --- | --- |
| Risico | Server-side request forgery |
| Aanvalstechniek | Proberen om slechte verzoeken naar de server te sturen via een kwetsbare applicatie. |
| Kans | Middel: nummer 10 van de OWASP 10. |
| Gevolg | Hoog: Ge Een aanvaller kan toegang krijgen tot interne netwerken of gevoelige gegevens door de kwetsbare server te misbruiken. |
|  |  |

* Alle gebruikersinput die wordt verwerkt via formulieren, wordt gevalideerd en gecontroleerd op het type variabele. Hiermee voorkom ik dat er onverwachte of malafide invoer wordt verwerkt.
* Voorbeeld:

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$input = $\_POST['input'] ?? null;

if (!is\_string($input)) {

die("Ongeldige invoer.");

}

// Verder verwerken van veilige invoer

}

# Zelfbeoordeling

WTIS-1: Voldoende/goed

WTIS-2:

* 2.2: Voldoende/goed

WTIS-3:

* 3.2 Goed

WTIS-4:

* 4.1: Voldoende
* 4.2: Goed
* 4.3: Voldoende

WTIS-5:

* 5.1: Voldoende
* 5.2: Voldoende

