Security analyse

Sil Gosker

**Versiebeheer**

|  |  |  |
| --- | --- | --- |
| **Wie** | **Wat** | **Wanneer** |
| Sil Gosker | Opzet van het document | 2025-02-10 |
| Sil Gosker | Analyseren van securityaspecten in UC1 | 2025-02-06 |
| Sil Gosker | Analyseren van securityaspecten in UC2 | 2025-02-08 |
| Sil Gosker | Analyseren van securityaspecten UC3 | 2025-02-25 |
| Sil Gosker | Analyseren van securityaspecten in UC4 | 2025-02-26 |
| Sil Gosker | Analyseren van securityaspecten in UC5 | 2025-02-28 |
| Sil Gosker | Analyseren van securityaspecten in UC6 | 2025-03-05 |

Contents

[2 Inleiding 4](#_Toc192144476)

[2.1 Probleemstelling 4](#_Toc192144477)

[2.2 Oplossing 4](#_Toc192144478)

[3 Threat modeling 5](#_Toc192144479)

[4 Use cases 6](#_Toc192144480)

[4.1 UC1: Bekijken profiel 6](#_Toc192144481)

[4.1.1 Evil user stories 6](#_Toc192144482)

[4.1.2 Assets 6](#_Toc192144483)

[4.2 UC2: Opnemen contact 7](#_Toc192144484)

[4.2.1 Evil user stories 7](#_Toc192144485)

[4.2.2 Assets 8](#_Toc192144486)

[4.3 UC3: inloggen 10](#_Toc192144487)

[4.3.1 Evil User stories 10](#_Toc192144488)

[4.3.2 Assets 10](#_Toc192144489)

[4.4 UC4: Registreren 12](#_Toc192144490)

[4.4.1 Evil user stories 12](#_Toc192144491)

[4.4.2 Assets 12](#_Toc192144492)

[4.5 UC5: Inzien planten 13](#_Toc192144493)

[4.5.1 Evil User stories 13](#_Toc192144494)

[4.5.2 Assets 13](#_Toc192144495)

[4.6 UC6: Aanmaken plant 14](#_Toc192144496)

[4.6.1 Evil User stories 14](#_Toc192144497)

[4.6.2 Assets 14](#_Toc192144498)

# Inleiding

Dit hoofdstuk omvat alle beveilingingsaspecten binnen de showcase. Dit zal gaan op basis van systeembrede aspecten (Thread modeling, security scans, etc) en use case specifieke aspecten (asset scan, vulnerability scan, etc.

## Probleemstelling

Stage aanbieders, werkgevers en recruiters zijn allemaal geïnteresseerd in nieuwe ontwikkelaars en willen oppervlakkig in contact met hen komen. Dit wordt in hoge mate vermoeilijkt door het feit dat er online geen verbinding gelegd kan worden met deze ontwikkelaars. Hierdoor wordt teruggevallen op sociale media platformen zoals LinkedIn, wat niet wenselijk is.

## Oplossing

De ontwikkelaar in kwestie zal een website ontwikkelen waarin duidelijk is wat hij kan en waar hij geïnteresseerd in is. Daarnaast moet een contactformulier geïntegreerd worden in de website waardoor oppervlakkig contact opgenomen kan worden.

# Threat modeling

Om systeem globaal weer te geven welke aspecten en eventuele veiligheidsissues komen kijken bij de huidige website, is een threat model gemaakt volgens de OWASP 2024 standaard. Per systeem en relatie zal gekeken worden naar de potentiële veiligheidsissues en hoe deze eventueel op te lossen zijn.

# Use cases

Iedere use case en zijn risico’s zullen besproken worden. Per use case zullen de “evil user stories”, assets en hun risico’s besproken worden.

Evil user stories zijn acties die een gebruiker wilt ondernemen in een voorgespecificeerd formaat. Dit formaat is als volgt: “Als [Rol] wil ik [Actie] zodat/omdat [Reden]”, waarbij de rol vervangen wordt door de corresponderende actor; de actie met de actie die hij uit wilt voeren; en de reden met waarom hij de actie wilt uitvoeren. Een voorbeeld van zo’n evil user story is “Als hacker wil ik een ReDoS attack uitvoeren zodat de server crasht”.

## UC1: Bekijken profiel

Dit hoofdstuk zal dieper ingaan op de verschillende potentiële veiligheidsissues binnen de use case UC1: Bekijken profiel. Voor de functionele werking kan naar het functioneel ontwerp gerefereerd worden.

### Evil user stories

#### Als data-scraper wil ik alle informatie van de ontwikkelaar van de webpagina halen zodat ik deze zelf kan publiceren

Probleem: Alle informatie op de kennismakingspagina kan d.m.v. webscraping gevonden worden (Owasp, 2020). Dit kan leiden tot herpublicaties van dezelfde informatie, wat uiteindelijk door duplicatieverlies kan leiden tot misinformatie.

Oplossing: Voorkomen dat webscrapers de website kunnen scrapen is praktisch onmogelijk. Goede scrapers imiteren browsers of zijn zelfs gebaseerd chromiumbrowsers zitten om te antibrowserdetectie te voorkomen. Voorkomen dat webscrapers de website kunnen vinden kan gedaan worden door dit te specificeren in de html-metadata.

Opgelost: Ja

### Assets

#### AS1: Ontwikkelaarsinformatie

De persoonlijke informatie van een ontwikkelaar staat op de hoofdpagina v/d website. In de onderstaande tabel kan exact gevonden worden welke specifieke informatie dit omvat en de waarde van die informatie, waarop een conclusie gevolgd wordt over de informatie van AS1.

|  |  |
| --- | --- |
| **Aspect** | **Waarde** |
| Foto’s | Laag/Nihil |
| Introductie | Laag/Nihil |
| Opsomming van vaardigheden | Laag/Nihil |

Alle aspecten van AS1 hebben een zéér lage waarde. Dit betekent dat als deze informatie misbruikt wordt op welke manier ook, er geen grote impact is op de ontwikkelaar, de website of anderen. Om deze reden zullen voorzorgsmaatregelen ook niet genomen worden.

## UC2: Opnemen contact

Dit hoofdstuk zal dieper ingaan op de verschillende potentiële veiligheidsissues binnen de use case UC1: Bekijken profiel. Voor de functionele werking kan naar het functioneel ontwerp gerefereerd worden.

### Evil user stories

#### Als een aanvaller wil ik scripts kunnen injecteren in de invoervelden zodat ik vertrouwelijke gegevens kan achterhalen.

Probleem: Aanvallers kunnen speciaal gemarkeerde waarden “injecteren” in invoervelden zodat ze code uit kunnen voeren op de website. Dit kan catastrofale gevolgen hebben, zoals het inlezen van cookies, bearer tokens, verschillende vormen van lokale opslag en andere informatie. Daarnaast kunnen ze, als de website andere kwetsbaarheden kent, misbruik maken van de rechten van de huidig-ingelogde gebruiker.

Oplossing: Alle invoervelden moeten “ontsmet” (sanitized) worden, en de waarde ervan mag niet zomaar in andere html-elementen op de website terugkomen. Dit kan gedaan worden door gebruik te maken van javascript-frameworks of libraries. Het huidig gebruikte framework (svelte) heeft dit standaard ingebouwd.

Opgelost: Ja

#### Als een aanvaller wil ik door lange invoer het geheugen kunnen overbelasten of een systeem crash kunnen veroorzaken.

Probleem: Aanvallers kunnen extensief lange invoer in een inputveld zetten. Wanneer dit verstuurd wordt naar de server, wordt deze geforceerd om deze lange invoer in het geheugen te laden, wat een crash kan veroorzaken.

Oplossing: Alle invoervelden moeten een maximale lengte bevatten en hier moet voor gevalideerd worden, zowel client-side als server-side.

Opgelost: Ja

#### Als aanvaller wil ik invalide data kunnen invoeren om het systeem te ontregelen.

Probleem: Javascript is een dynamically-typed programmeertaal. Dit betekent dat het toestaat om strings in een integere variabele te zetten, en andersom. Dit kan fouten veroorzaken, zowel client-side als server-side.

Oplossing: Alle data die verwerkt wordt op zowel de server als client moet gevalideerd worden op hun type, met een hoge preferentie op strongly-typed talen zoals C#.

Opgelost: Ja

#### Als aanvaller wil ik een ReDoS aanval kunnen uitvoeren zodat het systeem overbelast raakt en crasht.

Probleem: Aanvallers kunnen een e-mail adres invoeren met extensief veel korte domeinnamen. Dit kan bij verschillende soorten regex ervoor zorgen dat deze exponentieel recursief veel stackframes aanmaakt, wat in een stackoverflow kan resulteren (Owasp, 2024). Dit kan zelfs de server uitschakelen.

Oplossing: Een timeout op validatie d.m.v. regex kan ervoor zorgen dat de server niet genoeg tijd heeft om de stackframes te genereren die in een stackoverflow resulteren.

Opgelost: Ja

#### Als aanvaller wil ik de API zonder het website project aanroepen zodat ik het systeem kan overbelasten

Probleem: Aanvallers kunnen de api aanroepen zonder gebruik te maken van de website. Met gepersonaliseerde scripts kunnen ze op deze manier het systeem overbelasten door duizenden verzoeken tegelijk te sturen.

Oplossing: Een cors-policy op de API- en website toevoegen die voorkomt dat aanvragen verwerkt moeten worden (Owasp, 2020).

Opgelost: Ja

### Assets

#### AS2: Contactformulier

De persoonlijke informatie die een geïnteresseerde verplicht is in te vullen bij het opnemen van contact omvat de eerste asset van UC2. In de onderstaande tabel kan gevonden worden welke informatie dit omvat:

|  |  |
| --- | --- |
| **Aspect** | **Waarde** |
| Voornaam en achternaam | Medium |
| E-mail | Medium |
| Telefoonnummer | Medium |
| Onderwerp | Laag |
| Bericht | Medium |

Veel van de informatie is geclassificeerd als een “Medium”-waarde. Dit betekent dat, naast de veiligheidsmaatregelen besproken in 4.2.1 er geen actieve maatregelen genomen hoeven te worden.

#### AS3: API sleutels voor de mailing-API

Om gebruik te maken van een mailing-API zoals mailtrap, zijn API-sleutels nodig. In de onderstaande tabel kan gevonden worden welke sleutels nodig zijn en hun waarde/impact.

|  |  |
| --- | --- |
| **Aspect** | **Waarde** |
| Gebruikersnaam | Medium/Hoog |
| Wachtwoord | Hoog |

De gebruikersnaam/wachtwoord combinatie mag nooit vrijgegeven worden aan het publiek. Als deze wel vrijkomen, kan dit catastrofaal gevolgen hebben, waaronder het versturen én ontvangen van e-mails onder de naam van de ontwikkelaar.

Om dit te mitigeren zijn verschillende voorzorgsmaatregelen genomen, waaronder:

1. Deze asset mag niet in de broncode of het versiebeheer voorkomen.
2. Deze asset moet als omgevingsvariabele gebruikt worden, en versleuteld opgeslagen worden.

## UC3: inloggen

Dit hoofdstuk zal dieper ingaan op de verschillende potentiële veiligheidsissues binnen de use case UC3: inloggen. Voor de functionele werking kan naar het functioneel ontwerp gerefereerd worden.

### Evil User stories

#### Als aanvaller wil ik SQL injecties invoeren zodat ik de database kan uitlezen

Probleem: Aanvallers kunnen Specifieke SQL codes invoeren om het systeem te ontregelen. Het resultaat is dat een deel of zelfs de volledige database wordt blootgelegd.

Oplossing: Een bekende ORM gebruiken die SQL injecties mitigeert.

Opgelost: Ja

#### Als aanvaller wil ik e-mail adressen in kunnen voeren zodat ik weet of ze in de database bestaan

Probleem: Gebruikers kunnen onderzoeken of een e-mail adres in de database staat op basis van de foutmelding. Als deze “Geen gebruiker gevonden met dat e-mail adres” is, geeft dit informatie over de gebruiker bloot.

Oplossing: Een gegeneraliseerd bericht tonen waarbij niet duidelijk is of het account bestaat.

Opgelost: Ja

#### Als aanvaller wil ik een Long Password DoS attack uitvoeren zodat het systeem overbelast raakt

Probleem: Een aanvaller kan een zodanig lang wachtwoord invoeren, dat de gebruikte hashingmethode óók een extreem lange hash genereert. Het opslaan van zo’n hash in de database kan de database overbelasten, wat tot een Denial of Service volgt.

Oplossing: Niet alleen een minimale lengte, maar ook een maximale lengte op de wachtwoorden toepassen. OWASP geeft aan dat minimum maximum aantal karakters 64 moet zijn voor een wachtwoord.

Opgelost: Ja

### Assets

#### AS4: Inlogformulier

De persoonlijke informatie (email/wachtwoordcombinatie) omvat asset 4. In de onderstaande tabel kan gevonden worden welke informatie dit omvat:

|  |  |
| --- | --- |
| **Aspect** | **Waarde** |
| E-mail | Medium |
| Wachtwoord | Hoog |

De inlogggevens zijn een belangrijk aspect van de applicatie en moet zéér goed beveiligd worden, ondanks dat de individuele aspecten misschien niet een kritieke waarde hebben. Echter, naast de genomen maatregelen in 4.3.1 Evil User stories zijn geen andere maatregelen nodig. Propere hashing, versleuteld internetverkeer en gegeneraliseerde foutmeldingen zijn voldoende.

#### AS5: Private keys voor Bearer

Om bearer tokens te genereren is gebruik gemaakt van het Sha256 algoritme. Deze vereist een privé sleutel. In de onderstaande tabel kan de waarde van deze sleutels bekeken worden:

|  |  |
| --- | --- |
| **Aspect** | **Waarde** |
| Sleutel | Kritiek |
| Issuer | Medium |
| Audience | Medium |

Zoals aangetoond in de tabel is het beveiligen van deze sleutel van het hoogste belang. Als deze sleutel vrij gebruikt kan worden, kan iedereen zich voordoen op de applicatie die hij wilt zijn. Om dit te mitigeren moet deze sleutel beveiligd opgeslagen worden in een key-vault, om vervolgens als omgevingsvariabele gebruikt te worden.

## UC4: Registreren

Dit hoofdstuk zal dieper ingaan op de verschillende potentiële veiligheidsissues binnen de use case UC4: registreren. Voor de functionele werking kan naar het functioneel ontwerp gerefereerd worden.

### Evil user stories

#### Als aanvaller wil ik SQL injecties invoeren zodat ik de database kan uitlezen

Gemitigeerd in 4.3.1.1.

#### Als aanvaller wil ik e-mail adressen in kunnen voeren zodat ik weet of ze in de database bestaan

Probleem: Gebruikers kunnen onderzoeken of een e-mail adres in de database staat op basis van de foutmelding. Als deze “Geen gebruiker gevonden met dat e-mail adres” is, geeft dit informatie over de gebruiker bloot.

Oplossing: Captcha toevoegen bij het bevestigen van een registratie.

Opgelost: Nee.

### Assets

#### AS6: Registratieformulier

De persoonlijke informatie voor het definiëren van een account omvat asset 5. In de onderstaande tabel kan gevonden worden welke informatie dit omvat:

|  |  |
| --- | --- |
| **Aspect** | **Waarde** |
| E-mail | Medium |
| Wachtwoord | Hoog |

De inlogggevens zijn een belangrijk aspect van de applicatie en moet zéér goed beveiligd worden, ondanks dat de individuele aspecten misschien niet een kritieke waarde hebben. Echter, naast de genomen maatregelen in 4.3.1 Evil User stories zijn geen andere maatregelen nodig. Propere hashing, versleuteld internetverkeer en captcha zijn voldoende.

## UC5: Inzien planten

Dit hoofdstuk zal dieper ingaan op de verschillende potentiële veiligheidsissues binnen de use case UC5: inzien planten. Voor de functionele werking kan naar het functioneel ontwerp gerefereerd worden.

### Evil User stories

#### Als aanvaller wil ik de server spoofen zodat ik planten van andere gebruikers kan zien.

Probleem: Dit omvat een aanval waarbij foutieve gebruikers-identificatie naar de server wordt gestuurd. De gebruikers-identificatie is vormgegeven middels de Id- kolom van de Account-tabel (zie het technisch ontwerp voor meer informatie). Door een andere waarde mee te geven kunnen planten van andere gebruikers getoond worden.

Oplossing: De gebruikers-identificatie niet blootstellen aan de client.

Opgelost: Ja. Gebruikersidentificatie zit in de token, maar deze kan alleen decrypted worden door de server.

### Assets

Er zijn geen benoemingswaarde assets bevonden in deze use case.

## UC6: Aanmaken plant

Dit hoofdstuk zal dieper ingaan op de verschillende potentiële veiligheidsissues binnen de use case UC5: inzien planten. Voor de functionele werking kan naar het functioneel ontwerp gerefereerd worden.

### Evil User stories

Naast eerder genoemde evil user stories zijn geen andere benoemingswaardige user stories relevant tot deze use case.

### Assets

#### AS7: Inloggegevens plant

De inloggegevens van een plant (specifiek het database-identificatienummer en gegenereerde wachtwoord) omvat asset 7. In de onderstaande tabel staan de kwalitatief ingeschatte waarden.

|  |  |
| --- | --- |
| **Aspect** | **Waarde** |
| Database identificatienummer | Medium |
| Plant wachtwoord | Hoog |

Het wachtwoord van de plant wordt als “hoog” geschat, en dat is logisch. Als beide stukken informatie verkregen worden, kan zonder problemen verbinding worden gelegd met de server. Dit kan foutieve informatie veroorzaken, en andere hydrocomputers kunnen geen verbinding leggen voor die specifieke plant. Daarom wordt het wachtwoord gehasht in de database. Nadat het wachtwoord in plaintekst aan de gebruiker is getoond, kan deze niet meer teruggehaald worden.

Preferabel wordt er een “reset”- functionaliteit ingebouwd zodat de inloggegevens kunnen worden veranderd wanneer wenselijk.