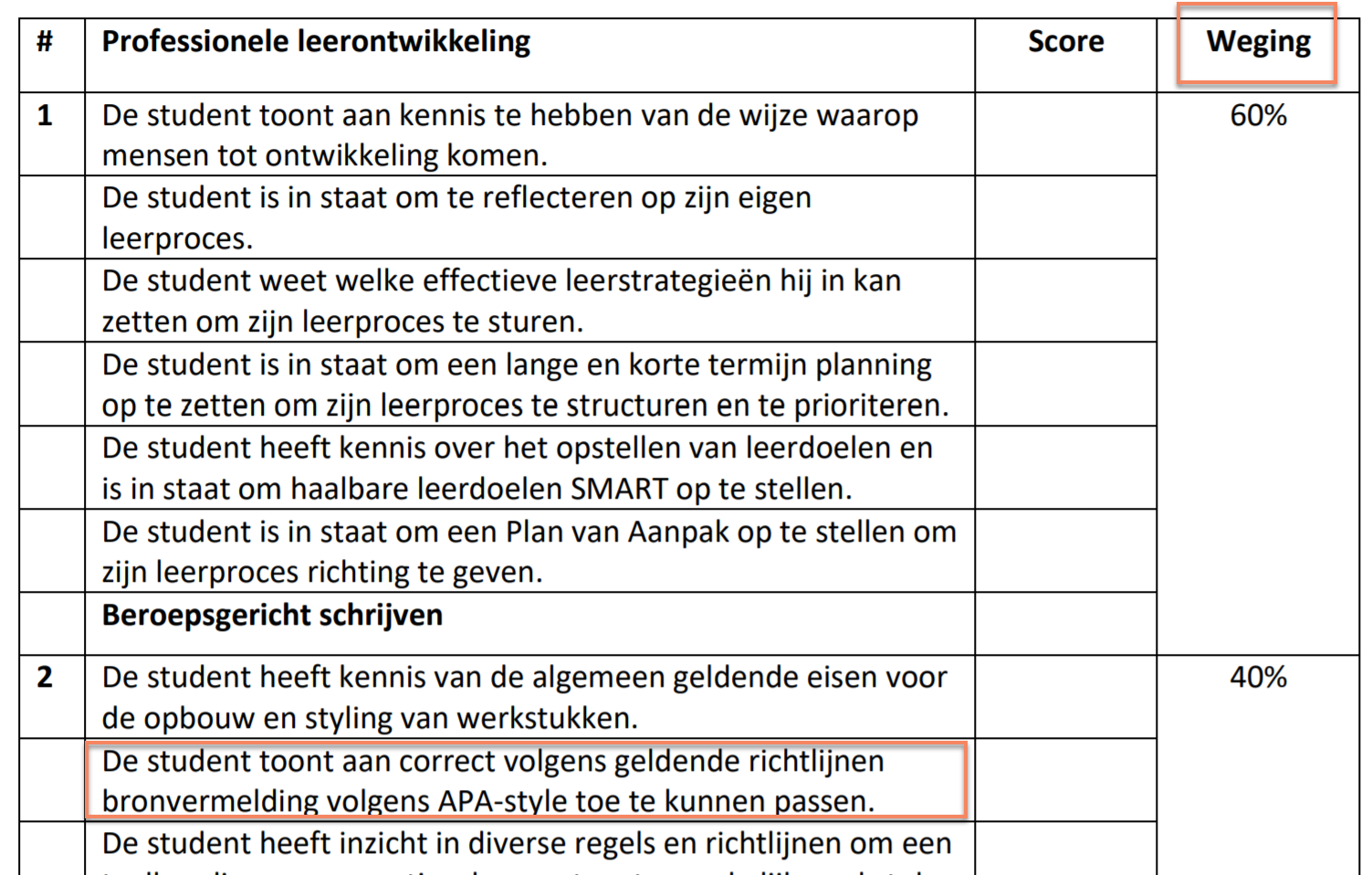
Risicoanalyse/Beveiligingsanalyse  
Esther Adjei Mensah  
Novi Hogeschool BV/Erik Mols

Ook al staat in de gegeven beoordelingscriteria niet vermeld dat er wordt beoordeeld op ‘het wel of niet gebruiken van de APA-conventie’ zoals hieronder bij een andere leerlijn, en werd hierover in de gegeven stof, in Edhub en tijdens de lessen geen uitleg over gegeven (waarschijnlijk omdat je daarop niet beoordeeld zou moeten worden), heb ik toch het initiatief genomen om even op te zoeken wat de APA is en deze toegepast.

Fig 1. *Beoordelingscriteria Leerlijn “Proffessionele ontwikkeling”.*

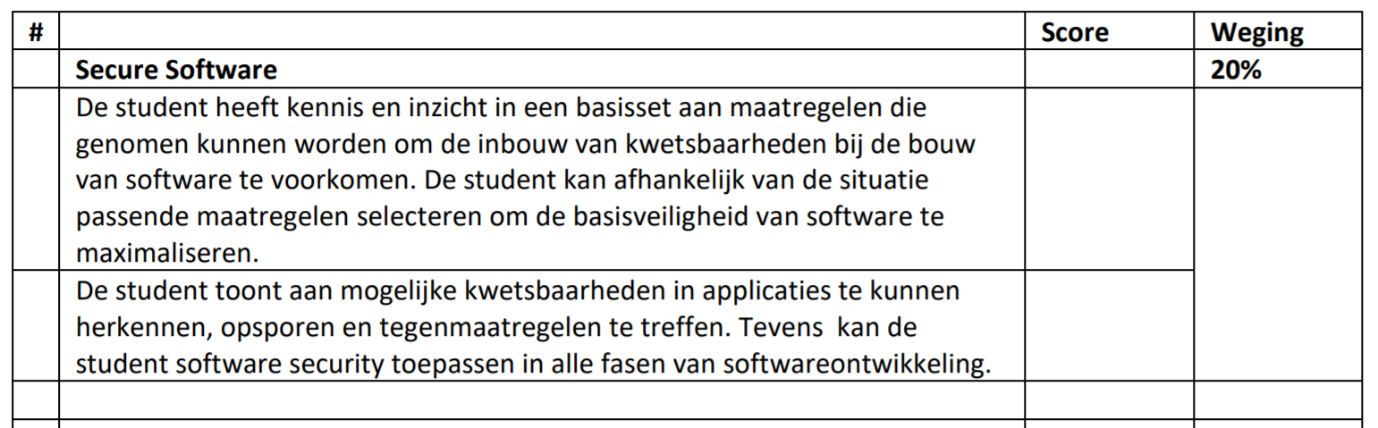


(Hogeschool NOVI B.V., z.d.)

Met dit document wil ik alleen in kaart brengen wat de mogelijke risico’s zijn gezien de gegeven kennis voor het maken van deze applicatie. Het betekent niet dat ik voor elke beveiligingsrisico een oplossing heb of zal toepassen en dat zul je in de realiteit bij elke software moeten afwegen. Het betekent ook niet dat als ik een beveiligingsrisico niet behandel, dat de doorzetting van de maak van de applicatie niet door zou gaan. In dit document

Verder breng ik in kaart m.b.v. de bekende dreigingen (STRIDE, OWASP Top-10) welke risico’s er bestaan en hoeveel impact deze zullen hebben als ze gebeuren.

Fig.2



(Hogeschool NOVI B.V., z.d.)

**Risicoanalyse/Beveiligingsanalyse**

De risicoanalyse heeft tot doel per project zwakheden in de beveiliging of opzet van de software te vinden en te onderkennen. ([Bron: Edhub, Grip op Secure Software, 4.3, alinea 1](https://edhub.novi.nl/study/courses/291/content/6725)) Hiervoor gebruik ik de uitgangspunten van STRIDE als aanleiding. Ook tijdens het ontwikkelen van de software houd ik hier rekening mee. Waarschijnlijk is het alleen DOS waar ik mij zorgen over hoef te maken als ik het project ga deployen.

Het eindresultaat van de risicoanalyse is een lijst met dreigingen die relevant worden geacht voor de IT-middelen binnen de scope en inzicht in de ernst van deze dreigingen. ([Bron: Edhub, Grip op Secure Software, 4.3, alinea 15](https://edhub.novi.nl/study/courses/291/content/6725)) Dit zijn de onderdelen met dreigingen die ik relevant acht voor de maak van deze software.

Lijst van dreigingen:

* Spoofing
* Tampering
* Repudiation
* Information disclosure
* Denial of Service (DoS)
* Elevation of privilege

Het doel van de risicoanalyse is het in een zo vroeg mogelijk stadium identificeren en begrijpen van risico’s en het benoemen van mitigerende beveiligingseisen. ([Bron: Edhub, Grip op Secure Software, 4.3, alinea 16](https://edhub.novi.nl/study/courses/291/content/6725)) \* (Hogeschool NOVI B.V., z.d.)

**Inschatting van dreigingen via STRIDE**De analysemethode STRIDE is ontwikkeld door Microsoft. Dit is een ‘threat assessment’. Er wordt een decompositie uitgevoerd, waarna per relevante component de gevoeligheid voor dreigingen wordt geanalyseerd. ([Bron: Edhub, Grip op Secure Software, 4.3, alinea](https://edhub.novi.nl/study/courses/291/content/6725) 9) (Hogeschool NOVI B.V., z.d.)

De naam STRIDE is een afkorting van de benamen van zes categorieën aan dreigingen, namelijk:

* Spoofing (misbruik van de gebruikersidentiteit, namelijk zich als een ander voordoen);
* Tampering (schending van de Integriteit);
* Repudiation (weerlegbaarheid);
* Information disclosure (schending van de privacy of het lekken van data);
* Denial of Service (DoS) (on-beschikbaarheid);
* Elevation of privilege (misbruik van bevoegdheden)

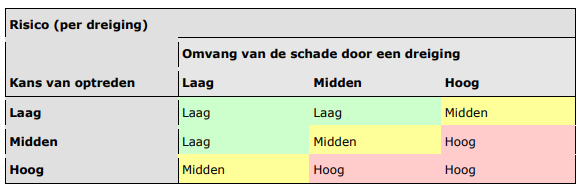
Op basis van deze dreigingen voer ik de risico/beveiligingsanalyse uit.

**Kwalificering Risico’s**

De kwalitatieve risicoanalyse gaat uit van scenario’s en situaties. Hierbij worden de kansen dat een dreiging werkelijkheid wordt ingeschat op basis van vuistregels en waarschijnlijkheid. De kwantitatieve risicoanalyse probeert op basis van risicowaardering te berekenen hoe groot de kans is dat een dreiging een incident wordt ([Bron: pinkelephant.nl Risicoanalyse](https://www.pinkelephant.nl/kennisbank/iso-27002-informatiebeveiliging/risicoanalyse/))

Voorbeeld:

Fig 2 kwalificering van dreigingen, volgens Edhub

******

Afbeelding bron: ([Edhub, Grip op Secure Software, 4.3](https://edhub.novi.nl/study/courses/291/content/6725)

(Hogeschool NOVI B.V., z.d.)

**De lijst met dreigingen:**

* Spoofing (misbruik van de gebruikersidentiteit, namelijk zich als een ander voordoen);
* Tampering (schending van de Integriteit);
* Repudiation (weerlegbaarheid);
* Information disclosure (schending van de privacy of het lekken van data);
* Denial of Service (DoS) (on-beschikbaarheid);
* Elevation of privilege (misbruik van bevoegdheden)

De bovengenoemde dreigementen kunne door middel van gebruik van het Spring Security Framework worden bestreden. Met uitzondering van Denial of service. In het geval van dit project wordt de applicatie alleen gedraaid op de local host van de beoordelaar, en/of AWS. Als dit het geval is regelt Amazon bescherming tegen Denial of service. Dit besef (DoS) is ook meegenomen in de kwalificering “kans van optreden”.

Er bestaat een kans dat een hacker een van de bovengenoemde dreigementen kan uitvoeren. Hoe groot is dan de impact op gebruikers of het bedrijf?  
Hieronder volgt dan de kwalificering voor de kans van optreden en de impact bij DIT project.

Impact 🡪

|  |  |  |  |
| --- | --- | --- | --- |
|  | Laag | Midden | Hoog |
| Hoog | Tampering  Elevation of privilege |  |  |
| Midden | Information Disclosure |  |  |
| Laag  Kans van optreden 🡪 | Denial of Service  Repudiation | Spoofing |  |

Dit baseer ik op de ernst van schade die kan worden verricht voor de opdrachtgever en de applicatie: \*Maar omdat ik nu niet meer met de opdrachtgever (Arjen Wiersma) in aanraking kan komen om nog meer aanvulling te geven, gebruik ik mijn eigen aannames voor de mogelijke gevaren voor de opdrachtgever, gebruikers en de applicate.  
Dit betreft alleen het stukje “impact … omdat”. Uiteraard als ik nog toegang had om de opdrachtgever te spreken, zou hier zijn citaten komen te staan.

**Spoofing  
Kans van optreden midden omdat:** (Userside) iedereen zich makkelijk als een ander kan voordoen door de informatie die hij invoert. In dit systeem vragen we niet naar gegevens om de échte identiteit van onze gebruikers te valideren.

(Bij de Bo-side is de kans van optreden laag omdat Backofficemedewerkers worden aangemaakt door de Admin, dus zij hebben niet de kans om gemanipuleerde data op te geven.)

**Impact laag omdat:** Deze informatie-systeem wordt gebruikt met als doel om als producer ontdekt te worden. Iemand die serieus in aanraking wil komen met het platenlabel Hexagon of Don Diablo zal zich niet voor een ander voordoen.

Ook is gevraagde informatie van de gebruikers niet erg gevoelig. Er wordt bijvoorbeeld niet gevraagd naar, geboortedatum, geboorteplaats, woonplaats, bankgegevens of andere gevoelige gegevens die crusiaal zijn in dit geval.

**Tampering   
Kans van optreden laag omdat:  
Impact hoog omdat:**

**Repudiation  
Kans van optreden laag omdat:  
Impact laag omdat:**

**Information Disclosure  
Kans van optreden laag omdat:  
Impact midden omdat:** De opgeslagen informatie van de gebruikers niet gevoelig of waardevol. Gegevens als: geboortedatum, geboorteplaats, woonplaats, bankgegevens, BSN of soortgelijk gevoelige data worden niet gevraagd. Waarmee het stelen van een identiteit niet succesvol kan worden verricht.

Mocht een hacker bij deze informatie komen heeft hij toegang tot de voornaam, achternaam, artiesten-naam, het emailadres/gebruikersnaam en de geuploadde demos. Over deze demo’s beschikken natuurlijk auteursrechten. Maar blijven “eigendom van Hexagon” zoals de opdrachtgever heeft aangegeven

Wachtwoorden zijn gehasht.

Een extra maatregel om de impact naar ‘laag’ te veranderen zou zijn om het emailadres, de achternaam én het adres naar de demo ge-encrypt op te slaan in de database. Maar zonder deze extra maatregel is de impact ‘midden’. De kans is klein dat dit in deze versie van het project gaat worden toegepast. Vandaar dat ik het nog steeds op midden laat.

Een ander ding om rekening mee te houden is de media. Als bekend wordt dat er een informatie-lek is kan dit mogelijk schade verrichten aan de naam van het bedrijf. Nogmaals dit hangt er vanaf belangrijk deze opgeslagen data is.

**DoS  
Kans van optreden laag omdat:** Er twee scenario’s zijn:  
A. Het project dient alleen gedraaid te worden op de localhost van de beoordelaar  
B. Het project wordt gehost op AWS.  
**Impact laag omdat:** Dit betekent dat het systeem tijdelijk niet meer te bereiken is. Als het goed is kan Amazon dit snel oplossen, en anders is het makkelijk om de software en de database naar een andere host te migreren. Voor de medewerkers en users is het dan een kwestie van wachten, maar er gaat geen informatie verloren.

**Elevation of priviledge  
Kans van optreden laag omdat:** Omdat je als developer met Spring Security goed kan aangeven welke rollen met welke privileges er zijn. Het Spring Security Framework is ook erg betrouwbaar. **Impact hoog omdat:**

**OWASP**

Ook al werd er GÉÉN uitleg gegeven over OWASP in Edhub en is dit woord niet één keer benoemd in de gegeven leerlijn “Secure Software” (ga maar na <https://edhub.novi.nl/study/learnpaths/236/courses>), heb ik toch heb ik het initiatief genomen om even op te zoeken wat het is, want ondanks dat dit nooit is behandeld, ben ik hier toch op beoordeeld voor het ontbreken van dit stukje informatie.

OWASP staat voor Open Web Application Security Project en is een open source-project rond computerbeveiliging. Individuen, scholen en bedrijven delen via dit platform informatie en technieken. (Wikipedia, 2019)

Er bestaat een zogeheten OWASP top-10 die erg hot blijkt te zijn. Eind 2017 heeft het Owasp een nieuwe top 10 gepubliceerd van de grootste en meest voorkomende risico's binnen webapplicaties. Deze top 10 is een belangrijk hulpmiddel voor webontwikkelaars om kwetsbaarheden te identificeren en te voorkomen.

Het eerste risico is dezelfde als in 2013 en 2010: **injectie**. Injectie is een algemene term voor het risico dat ontstaat wanneer commando's in een instructietaal (bijvoorbeeld SQL, Bash of LDAP) onveilig gemixt worden met gebruikersinvoer. Wanneer een aanvaller tekens met een speciale betekenis (‘metakarakters’) invoert, kan deze vervolgens de betekenis van dit commando aanpassen en vaak ernstige schade aanrichten. Het bekendste voorbeeld van zo'n aanval is SQL-injectie, waarbij een aanvaller eigen queries kan laten uitvoeren en daarmee de inhoud van de database kan uitlezen of wijzigen. Hoewel SQL-injectie al bijna twintig jaar een groot probleem is, komen we het tijdens beveiligingsonderzoeken steeds minder vaak tegen. Ontwikkelaars zijn tegenwoordig goed op de hoogte van deze beruchte aanval en web-frameworks zijn er standaard op ingericht om dit risico te vermijden. Dat geldt echter niet persé voor NoSQL-injectie.

Het tweede risico is **authenticatiefouten**: voorkomen dat aanvallers andermans account overnemen blijft in het algemeen ingewikkeld. Trends zoals stateless sessiebeheer, single-sign on en authenticatiemethoden die ook geschikt zijn voor apps, introduceren nieuwe risico's, waar ontwikkelaars mogelijk minder bekend mee zijn.

Het derde risico is het **lekken van gevoelige data**, bijvoorbeeld **door een gebrek aan versleuteling of (sterke) gebruikersauthenticatie**. Ook worden tegenwoordig gevoelige gegevens nog vaak, per ongeluk, publiek toegankelijk gemaakt. Een pluspunt is wel dat het gebruik van transportversleuteling, door middel van https, enorm is toegenomen.

Het vierde risico op de lijst is: **het toestaan van externe xml-entiteiten**. Dit is een oud probleem dat aanvallers toestaat bestanden op de server (en het netwerk achter de firewall) uit te lezen door een, bij nader inzien niet zo handige, feature uit te buiten in (verouderde) software die xml-berichten uitleest.

Het vijfde risico is **falende toegangscontroles**. Een klassiek probleem dat bij moderne webtechnologieën ook erg vaak voorkomt. Het ontbreken van autorisatiecontroles is ook een typische kwetsbaarheid dat lastig is om op te testen, of om te vinden met automatische tools.

Het zesde risico luidt: **configuratiefouten**. Software is vaak standaard ingesteld zodat het makkelijk in gebruik te nemen is, maar niet persé op een manier die veilig is. Er wordt aangeraden om ervoor te zorgen dat de belangrijke security-functies altijd aan staan, er geen gebruik wordt gemaakt van standaardwachtwoorden en debug-functies uit te zetten op de productieomgeving.

Het zevende risico is **cross-site-scripting**, het overnemen van andermans browsersessie door scripts in een webpagina te injecteren, is flink gedegradeerd: van nummer drie in 2013 tot nummer zeven in de nieuwste editie. Deze kwetsbaarheid kan nog steeds catastrofaal zijn, maar is vanwege mitigaties van web-frameworks moeilijker uit te buiten dan voorheen. Dit soort beveiligingsmaatregelen worden echter vaak expliciet uitgeschakeld wanneer ze in de weg zitten, en ze beschermen ook niet tegen alle mogelijke vormen van cross-site-scripting.

De top 10 wordt afgesloten met **respectievelijk onveilige deserialisatie**: Het mogelijk maken voor een aanvaller om een getransporteerd programmeertaal-object aan te passen, waardoor de server overgenomen kan worden.  
Het **gebruik van software met bekende kwetsbaarheden** (tegenwoordig gemakkelijk uit te buiten met hacker-tools waarmee op verouderde software gescand kan worden).  
En **onvoldoende logging en monitoring**. Het laatste risico erkent dat je niet kunt aannemen dat preventieve maatregelen afdoende zijn: het is ook belangrijk om aanvalspogingen te kunnen detecteren, adequaat erop te kunnen reageren, en een plan te hebben voor wanneer het misgaat.

(Tervoort, 2018)

**Beveiligingsmaatregelen**

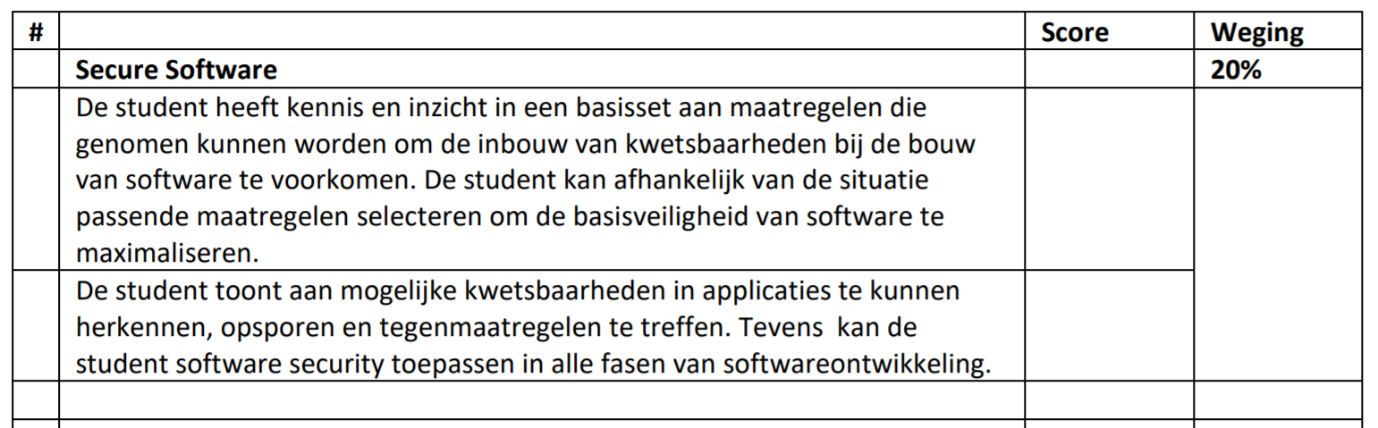
Dit zijn de beveiligingsmaatregelen die ik verwacht te gebruiken in de applicatie. Dit baseer ik op de opgedane kennis die ik heb gedaan bij het bouwen van andere projecten:

* Bescherming tegen csrf
* Input validatie in max aantal, elk naar wat de developer verstandig lijkt te zijn om de applicatie niet te laten crashen.
* Rollen en Autorisatie
* Authenticatie
* Sessiebeheer
* Bescherming tegen spoofing van een emailadres door een bevestigings-email met code te sturen.
* Bescherming tegen crossite scripting

Dit document was een mix van de stof in Edhub over de risico-analyse, extra informatie over Owasp en het kunnen aantonen dat ik kennis en inzicht heb in een basisset maatregelen.

Ik hoop met dit document te hebben aangetoond dat ik kennis en inzicht heb in een basisset aan maatregelen die genomen kunnen worden om de inbouw van kwetsbaarheden bij de bouw van software te voorkomen.

Tijdens de bouw van de applicatie schrijf ik een verantwoordingsdocument met de toegapaste security in alle fasen van de softwareontwikkeling: Frontend, backend en de opgeslagen data.  
Ik toon aan de mogelijke kwetsbaarheden te kunnen herkennen, opsporen en hoe ik tegenmaatregelen tref.



**Einde Risico-analyse**

# Bronnenlijst

Hogeschool NOVI B.V. (z.d.). *Eindopdracht Full Stack Developer Beoordelingscriteria.* Opgehaald van edhub.novi.nl: https://edhub.novi.nl/study/learnpaths/252/documents

Hogeschool NOVI B.V. (z.d.). *Proffessionele ontwikkeling.* Opgehaald van edhub.novi.nl: https://edhub.novi.nl/study/learnpaths/247/documents

Hogeschool NOVI B.V. (z.d.). *Risicoanalyse.* Opgehaald van edhub.novi.nl: https://edhub.novi.nl/study/courses/291/content/6725

Tervoort, T. (2018, 01 19). *Owasp top 10 van web-risico’s is vernieuwd*. Opgehaald van computable.nl: https://www.computable.nl/artikel/expertverslag/security/6280786/4573232/owasp-top-10-van-web-risicos-is-vernieuwd.html

Wikipedia. (2019, 11 2). *OWASP*. Opgehaald van Wikipedia.org: https://nl.wikipedia.org/wiki/OWASP