OWASP Top 10

Spots

Door: Paul Evers (3496783)

## Versies

|  |  |  |  |
| --- | --- | --- | --- |
| **Versie** | **Datum** | **Auteurs** | **Wijzigingen** |
| 1 | 26-5-2021 | Paul Evers | Eerste versie |

## 

## Introductie

De OWASP Top 10 is een lijst die de 10 meest voorkomende security risico’s definiëren in web applicaties. In dit document ga ik voor ieder punt kijken wat deze inhoudt en hoe deze van toepassing is op Spots. Daarbij wordt een schatting gemaakt van het huidige risiconiveau voor het desbetreffende punt.

Het risiconiveau wordt als volgt gedefinieerd:

|  |  |  |  |
| --- | --- | --- | --- |
| Laag | Gemiddeld | Hoog | Niet van toepassing |

## 

## 

## **1. Injection**

Met injectie kunnen door middel van bijvoorbeeld SQL of NoSQL, queries utigevoerd worden die de ontwikkelaar niet bedoeld heeft. Zo kan er bijvoorbeeld afgeschermde data opgehaald worden, data gemanipuleerd of verwijderd worden.

**In Spots**

Binnen Spots wordt gebruik gemaakt van de NoSQL database MongoDB. Voor het uitvoeren van query's naar deze database wordt gebruik gemaakt van Spring Data. Hierbij wordt gebruik gemaakt van Object-relational mapping (ORM). Hierdoor worden er geen zelfgeschreven queries uitgevoerd binnen Spots. Het zou kunnen dat bij het toevoegen van functionaliteit dit toch wordt gedaan. Indien er handmatig queries worden uitgevoerd moet er hier rekening gehouden mee worden.

**Huidig gevaar:**

|  |
| --- |
|  |

## **2. Broken Authentication**

Als er sprake is van Broken Authentication kan er ongeoorloofd ingelogd worden op een account wat niet tot die persoon behoort. Dit kan komen door het bemachtigen van de inloggegevens or de acces tokens.

**In Spots**

Spots maakt gebruik van Google Firebase Authentication wat zich berust op OAuth 2.0. Firebase Authentication is een vertrouwde techniek die door de hele industrie gebruik wordt. Indien er toch een token bemachtigd wordt is deze maar tijdelijk bruikbaar. Iedere token heeft een relatief korte levensduur van een paar uur.

Daarnaast worden gevoelige gegevens, de email en wachtwoord, beheerd door Google, die erom bekend staan een grote veiligheid te garanderen.

Ook zijn enkele endpoints niet beschikbaar voor alle gebruikers omdat er gebruik wordt gemaakt van rollen. Hierdoor kunnen bepaalde endpoints alleen bereikt worden als de persoon daarvoor ook de juiste rol heeft.

**Huidig gevaar:**

|  |
| --- |
|  |

## **3. Sensitive Data Exposure**

Gevoelige informatie zoals wachtwoorden en persoonsgegevens mogen niet in de verkeerde handen vallen. Daarnaast zijn er ook wetten zoals de AVG die gebruikers moet garanderen dat er goed met hun gegevens omgegaan wordt. Als deze gegevens vrijkomen kan dit een inbreuk op de privacy van gebruikers zijn.

**In Spots**

Binnen Spots wordt tot nu toe geen gevoelige informatie opgeslagen. De enige gevoelige informatie die naar de server gestuurd wordt is een email en wachtwoord om een account te maken. Dit kan beveiligd worden door een SSL verbinding op te zetten naar de server.

Daarnaast kan in de toekomst, indien er gevoelige informatie opgeslagen gaat worden, gebruik gemaakt worden van encryptie.

**Huidig gevaar:**

|  |
| --- |
|  |

## **4. XML External Entities**

XML kan gebruikt worden voor het manipuleren van de configuratie van systemen. Zo kun je interne bestanden bemachtigen, porten scannen, en ongewenste code uitvoeren op de systemen.

**In Spots**

Binnen Spots wordt geen gebruik gemaakt van XML en daar zijn ook geen plannen voor.

**Huidig gevaar:**

|  |
| --- |
|  |

## **5. Broken Acces Control**

Het kan voorkomen dat gebruikers wel toegang tot een systeem mogen hebben maar per ongeluk daarbij ook toegang krijgen tot systemen waar geen toegang voor is. Denk hierbij aan bijvoorbeeld de gegevens van andere personen.

**In Spots**

Zoals besproken bij *2. Broken Authentication* wordt er gebruik gemaakt van een rollensysteem. Dit zorgt ervoor dat gebruikers niet bij data en functies kunnen komen die niet voor hen bedoeld zijn. Zo kunnen gebruikers bijvoorbeeld alleen verwijderd worden door een gebruiker met de *admin* rol.

Het ophalen van gebruikersgegevens is ook alleen maar mogelijk voor de persoon die is ingelogd met die gegevens door de token die daarbij wordt meegestuurd.

Als nieuwe functionaliteiten worden toegevoegd moet er worden nagedacht over wie toegang tot deze functionaliteiten mag hebben, en hoe ze beschermd kunnen worden.

**Huidig gevaar:**

|  |
| --- |
|  |

## **6. Security Misconfiguration**

Het correct configureren van systemen kan vaak over het hoofd worden gezien. Denk hierbij aan het zetten van permissies, een productie-versie uitrollen, etc. Maar ook het niet tijdig updaten van libraries, frameworks, en besturingssystemen kan gevoeligheden bloot laten liggen.

**In Spots**

Dit is een reëel probleem dat zich kan voordoen binnen Spots. Er wordt gebruik gemaakt van CI/CD waarbij beter te garanderen is dat de correcte configuratie wordt meegegeven. Toch kunnen er nog steeds fouten over het hoofd gezien worden.

Binnen de projecten wordt gebruik gemaakt van Gradle als dependency manager. Hiermee kunnen dependencies makkelijk geupdate worden naar de laatste veiligheidsupdates.

**Huidig gevaar:**

|  |
| --- |
|  |

## **7. Cross-Site Scripting**

Cross-Site Scripting is het ongeoorloofd inbrengen van kwaadaardige code op een website.

**In Spots**

Het is de bedoeling dat Spots een Flutter app wordt waarbij XSS niet mogelijk is. Voor het management portal wordt een Angular applicatie gebouwd. Angular heeft verschillende beveiligingsmechanismen ingebouwd om XSS te vermijden. Daarnaast zal er opgepast moeten worden dat er geen onnodige *innerHtml* elementen gebruikt worden.

**Huidig gevaar:**

|  |
| --- |
|  |

## **8. Insecure Deserialization**

Bij het onveilig deserializen van objecten kan ongewenst gedrag plaatsvinden of ongewenste code uitgevoerd worden.

**In Spots**

Er wordt binnen Spots gebruik gemaakt van vertrouwde, veilige, en veelgebruikte deserializers zoals Jackson JSON. Als er data wordt gestuurd die niet overeenkomt met de bekende klassen vanuit het systeem zal het niet uitgevoerd worden.

**Huidig gevaar:**

|  |
| --- |
|  |

## **9. Using Components with Known Vulnerabilities**

Er bestaan frameworks, libraries, componenten en besturingssystemen die niet veilig en verouderd zijn. Vooral als ze deprecated zijn en niet geupdate kunnen worden is het niet slim deze te gebruiken.

**In Spots**

Voor de backend worden bekende open source libraries en frameworks gebruikt. Voornamelijk Spring. Er wordt gebruikt gemaakt van Spring Boot, Spring Gateway, Spring Eureka, Prometheus, Grafana, MongoDB, RabbitMQ en meer. Deze projecten hebben allemaal een grote community support achter zich en zijn open source.

Voor de front-end wordt gebruik gemaakt van Angular en Flutter, ook deze zijn beiden erg vertrouwd.

Zodra een nieuw component wordt toegevoegd is het belangrijk om te controleren of deze veilig is. Ook moeten oudere componenten in de gaten worden gehouden dat ze nog steeds veilig zijn.

**Huidig gevaar:**

|  |
| --- |
|  |

## **10. Insufficient Logging & Monitoring**

Als er niet genoeg of goed gemonitord en gelogd worden kunnen problemen binnen het systeem over het hoofd worden gezien. Om fouten zo snel mogelijk te kunnen opsporen, en dus oplossen, is het belangrijk een goed overzicht van de staat van je systeem te hebben.

**In Spots**

Er wordt met behulp van Prometheus en Grafana bijgehouden naar de performance van de back-end. Echter er is nog niet genoeg logging voor problemen die kunnen plaatsvinden. Er is ook geen makkelijke manier om nu logs te bekijken.

**Huidig gevaar:**

|  |
| --- |
|  |