Research OWASP



Gemaakt door:  
Sharony Braam  
DB01

Contents

[Probleemanalyse 3](#_Toc61944642)

[Context 3](#_Toc61944643)

[Doelgroep 3](#_Toc61944644)

[Doel 3](#_Toc61944645)

[Onderzoeksvragen 3](#_Toc61944646)

[Hoofdvraag 3](#_Toc61944647)

[Deelvragen 3](#_Toc61944648)

[Onderzoeksmethoden en strategieën 3](#_Toc61944649)

[Library – Product analysis 3](#_Toc61944650)

[Stepping stone – Risk Analysis 4](#_Toc61944651)

[Library – Best, good & bad practice 5](#_Toc61944652)

[Conclusie 7](#_Toc61944653)

[Deelvragen 7](#_Toc61944654)

[Wat is het OWASP Top 10? 7](#_Toc61944655)

[Aan welke punten zou GamingLads moeten voldoen? 7](#_Toc61944656)

[Hoe kan er worden voldaan aan de punten waaraan GamingLads nog niet voldoet? 7](#_Toc61944657)

[Hoofdvraag 8](#_Toc61944658)

[Hoe kan het project GamingLads voldoende beveiligd worden? 8](#_Toc61944659)

[Bronnen 8](#_Toc61944660)

# Probleemanalyse

### Context

De context voor dit onderzoek bevat het individueel project in semester drie tijdens de studie HBO-ICT te Fontys in Eindhoven. Het project heet GamingLads en dient de mensen bij elkaar te brengen d.m.v. games. Op deze manier wordt het vinden van een game-partner versimpeld en versneld.

### Doelgroep

De doelgroep voor dit project bestaat uit één groep, namelijk de klanten die gebruik gaan maken van het GamingLads project.

### Doel

Het doel van dit onderzoek is te onderzoeken hoe GamingLads beter beveiligd kan worden.

# Onderzoeksvragen

## Hoofdvraag

Hoe kan het project GamingLads voldoende beveiligd worden?

### Deelvragen

1. Wat is OWASP Top 10?
2. Aan welke punten zou GamingLads moeten voldoen?
3. Hoe kan er worden voldaan aan de punten waaraan GamingLads nog niet voldoet?

Voor deelvraag één wordt gebruik gemaakt van library – product analysis. Deze methode zorgt ervoor dat het makkelijk en snel op te zoeken is. Voor deelvraag twee wordt gebruik gemaakt van de steppingstone Risk Analysis. Deze methode helpt bij het ondervinden van risico’s en hoe groot die zijn. Het testrapport heeft hierbij geholpen. Dit testrapport is te vinden in dezelfde folder. Daarna in deelvraag drie wordt op basis van library – best, good & bad practices een manier gevonden om de eventuele risico’s die er zijn op te lossen.

# Onderzoeksmethoden en strategieën

## Library – Product analysis

OWASP staat voor Open Web Application Security Project. Het is een internationaal bedrijf dat zich focust op het ontwikkelen van applicaties die veilig zijn. Om die reden hebben zij een lijst gemaakt van de tien meest voorkomende beveiligingsfouten in een project. Zij hebben veel documentatie en tools om andere bedrijven te helpen met het beveiligen. De lijst bestaat uit:

1. Injection, bijna elke source van data kan geinject worden. Er is een inject flaw als een aanvaller data kan versturen naar iets dat het moet ontvangen. Dat kan eindigen in data verlies, corruptie en zelf het uitsluiten van toegang. Het kan soms leiden tot het totaal overnemen van een host.
2. Broken Authentication, aanvallers kunnen toegang krijgen tot enorm veel gebruikte usernames en passwords voor o.a een automatische brute force. Ook sessie management aanvallen worden veel gebruikt. Door dit kan er identiteitsfraude worden gepleegd of kunnen gegevens naar buiten worden gebracht.
3. Sensitive Data Exposure, veel applicaties doen niet genoeg om hun gevoelige data te beschermen zoals geldzaken, gezondheidszorg en meer. Aanvallers kunnen deze data stelen of aanpassen om o.a. identiteitsfraude te plegen.
4. XML External Entities (XXE), veel oude of slecht geimplementeerde XML processors hebben entity references in XML documenten. Het kan zorgen voor een internal port scan of een ontkenning van service aanvallen.
5. Broken Access Control, regels op gebruikers zijn vaak niet goed geforceerd. Aanvallers kunnen een fout vinden in dat systeem en krijgen toegang tot data en dat zij als gebruiker niet horen in te zien.
6. Security Misconfiguration, dit is het meest voorkomend probleem. Misgeconfigureerde http headers of error messages met gevoelige data. De security moet bijgehouden worden, netzoals als frameworks.
7. Cross-Site Scripting XSS, dit gebeurd wanneer er onbekende data in een nieuwe webpagina komt wanneer er o.a. geen validatie is gedaan. Hierdoor kunnen aanvallers scripts rdraaien die ervoor zorgt dat het slachtoffer zijn/haar gebruikers sessie wordt gehijackt of verstuurd wordt naar een kwaadaardige website.
8. Insecure Deserialization, dit kan leiden tot remote code execution. Zelfs als dit niet gebeurd kunnen er o.a. nog injection attacks worden uitgevoerd.
9. Using Components with Known Vulnerabilities, als er een fout zit in een component moet die eerst opgelost worden. Als dit niet gebeurd kan een aanval leiden tot data verlies of zelfs overname van de server.
10. Insufficient Logging & Monitoring, een breach wordt pas na tweehonderd dagen ontdekt. Het bedrijf zelf moet het product meer in de gaten houden.

## Stepping stone – Risk Analysis

Hier wordt gekeken naar het project GamingLads en hoe ver het aan de punten voldoet. Hier is gebruik gemaakt van het testrapport. Daarna wordt gekeken welke punten nu belangrijk zijn en welke later pas van toepassing zijn.

1. Injection. Alle beans worden geautowired d.m.v. de constructor. Login wordt gecheckt. Andere services die op dit moment niet gebruikt worden, zijn wel vatbaar.
2. Broken Authentication. Er zit nog geen checks op de passwords bij sign in. De passwords zijn ook nog niet hashed. Tokens moeten nog verder worden geimplementeerd. Hier zit dus nog een groot risico.
3. Sensitive Data Exposure. Zoals benoemd bij punt twee worden de passwords nog niet gehasht. Dit is een groot risico.
4. XML External Entities (XXE). De gegevens van de database staan in een xml file. Dit is een groot risico omdat er een ander xml file kan worden verstuurd en dan kan de data in verkeerde handen vallen.
5. Broken Access Control. Op dit moment hebben de rollen nog geen functies. Voor production is dit ook een groot risico. Het kan er namelijk voor zorgen dat alle data verloren gaat of wordt misbruikt.
6. Security Misconfiguration. Voor production moet er nog wat aangepast worden in spring security, maar dit is niet het grootste risico.
7. Cross-Site Scripting (XSS). In Vue is er geen automatische bescherming van XSS. Dit is iets wat in de toekomst gerealiseerd moet worden.
8. Insecure Deserialization. Iedereen die inlogt wordt gecheckt en krijgt daarna een JWT mee. Hier zouden nog wel wat dingen fout kunnen gaan.
9. Using Components with Known Vulnerabilities. Dit is in punt één benoemd. Er zijn een aantal services die op dit moment niet worden gebruikt. Als die gebruikt gaan worden, moeten ze eerst goed getest en geupdate worden. Voor nu is het dus nog geen risico.
10. Insufficient Logging & Monitoring. De logging van het gaminglads project is matig. Als het dus in production gaat, moet er nog wat aan veranderen, maar voor nu is het voldoende. Monitoring is alleen van toepassing als het in production gaat.

|  |  |
| --- | --- |
| **Wat?** | **Voldoet het aan de eisen?** |
| Injection | Nee |
| Broken Authentication | Nee |
| Sensitive Data Exposure | Nee |
| XML External Entities (XXE) | Nee |
| Broken Access Control | Nee |
| Security Misconfiguration | Nee |
| Cross-Site Scripting XSS | Nee |
| Insecure Deserialization | Nee |
| Using Components with Known Vulnerabilities | Nee |
| Insufficient Logging & Monitoring | Nee |

Zoals duidelijk te zien is, is er nog veel te verbeteren in het GamingLads project. Aangezien de niet-gebruikte services op dit moment geen risico vormen worden deze achterwegen gelaten. Dit houdt in dat punt twee, drie, zeven, acht en tien de belangrijkste zijn. Vier wordt achterwegen gelaten omdat het vooral een probleem is met legacy applicaties. Vijf wordt achterwegen gelaten omdat er nog geen admin portal is, punt zes heeft ook te maken met de rollen en punt één en negen met de ongebruikte services.

## Library – Best, good & bad practice

|  |  |  |
| --- | --- | --- |
| **Wat?** | **Hoe is het op te lossen?** | **Wat is toepasbaar op GamingLads** |
| Broken Authentication | * Gebruik waar mogelijk multifactor authenticatie om o.a brute force en het hergebruiken van gestolen credentials tegen te gaan. * Deploy niet met standaard credentials, vooral voor de admin gebruikers. * Implementeer een slecht-wachtwoord check met bijv. de top 10000 slechtste wachtwoorden. * Zet een minimaal wachtwoord-lengte en complexiteit. * Zorg dat alles met authentication en authorization veilig is voor account enumeration attacks * Limiteer of delay failed login attempts. Log alle data ervan en geef een alert aan administrators als er een aanval wordt gedetecteerd. * Gebruik een server-side, secure, built-in session dat genereert een nieuw willekeurig session ID na een login. Het ID mag niet in de url staan maar moet veilig worden opgeslagen en ongeldig worden gemaakt na een logout, afwezigheid en een timeout. | Alle punten zijn toepasbaar op GamingLads, maar punt twee, vier en punt zes zijn voldoende. |
| Sensitive Data Exposure | * Check data dat geprocessed, opgeslagen of verzonden wordt door een applicatie. Identificeer welke data gevoelig is d.m.v. privacy rechten en de wet. * Zet controles per check. * Sla geen gevoelige data op dat niet opgeslagen hoeft worden. Zodra het kan, verwijder het. * Alle gevoelige data moet encrypt worden. * Gebruik een goede key management. * Zet cashing uit voor het opslaan van gevoelige data. * Sla wachtwoorden op met een goede hashing zoals Argon2, scrypt, bcrypt of PBKDF2. * Verifieer per onderdeel het effect van de configuratie | Alle punten zijn toepasbaar op GamingLads, maar punt één, twee, drie, vier en zeven zijn voldoende. |
| Cross-Site Scripting XSS | * Gebruik frameworks die automatisch XSS ontwijken zoals React JS. Doe onderzoek naar elk framework en hun beveiliging. * Leer data escape technieken en voeg die toe. * Leer DOM based XXS prevention en voeg die toe. * Zet Content Security Policy aan voor beveiliging tegen XSS. | Alle punten zijn toepasbaar op GamingLads, maar die gaan vrij diep. Punt één is voor nu voldoende. |
| Insecure Deserialization | * Implementeer checks zoals digitale handtekeningen. * Zet stricte types tijdens een deserialization voordat een object gecreeerd wordt. Het is mogelijk om hier langs te komen dus leg hier niet alle vertrouwen in. * Deserialization kan het best gedaan worden in een lage privilege omgeving. * Log deserializations die falen, zoals waar de incoming type is niet gelijk aan het verwachte type. * Zet restricties en monitor inkomende en uitkomende connecties van containers of servers die deserializen. * Monitor het deserialization, zend een alert als een user constant bezig is met deserializen. | Alle punten zijn toepasbaar op GamingLads, maar punt twee, vier en vijf zijn voldoende. |
| Insufficient Logging & Monitoring | * Alle authorization requests die falen worden met genoeg gebruiker context gelogt, zodat de verdachte accounts eruit gefilterd kunnen worden. * Zorg dat de logs in een format is dat makkelijk kan worden opgeslagen in log management oplossingen. * Zorg voor effectieve monitoring en alerts. * Zorg voor een recovery plan zoals NIST 800-61 rev 2 of verdere versies. | Alle punten zijn toepasbaar op GamingLads maar punt één en twee zijn voor nu voldoende. |

# Conclusie

## Deelvragen

### Wat is het OWASP Top 10?

OWASP staat voor Open Web Application Security Project. Zij zijn een internationaal bedrijf dat zich focust op het ontwikkelen van applicaties die veilig zijn. Om die reden hebben zij een lijst gemaakt van de tien meest voorkomende beveiligingsfouten in een project. De lijst bestaat uit:

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting XSS
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring

### Aan welke punten zou GamingLads moeten voldoen?

Punt twee, drie, zeven, acht en tien zijn de belangrijkste. Vier wordt achterwegen gelaten omdat het vooral een probleem is met legacy applicaties. Vijf wordt achterwegen gelaten omdat er nog geen admin portal is, punt zes heeft ook te maken met de rollen en punt één en negen met de ongebruikte services.

### Hoe kan er worden voldaan aan de punten waaraan GamingLads nog niet voldoet?

|  |  |
| --- | --- |
| **Wat?** | **Hoe is het op te lossen?** |
| Broken Authentication | * Deploy niet met standaard credentials, vooral voor de admin gebruikers. * Zet een minimaal wachtwoord-lengte en complexiteit. * Limiteer of delay failed login attempts. Log alle data ervan en geef een alert aan administrators als er een aanval wordt gedetecteerd. |
| Sensitive Data Exposure | * Check data dat geprocessed, opgeslagen of verzonden wordt door een applicatie. Identificeer welke data gevoelig is d.m.v. privacy rechten en de wet. * Zet controles per check. * Sla geen gevoelige data op dat niet opgeslagen hoeft worden. Zodra het kan, verwijder het. * Alle gevoelige data moet encrypt worden. * Sla wachtwoorden op met een goede hashing zoals Argon2, scrypt, bcrypt of PBKDF2. |
| Cross-Site Scripting XSS | * Gebruik frameworks die automatisch XSS ontwijken zoals React JS. Doe onderzoek naar elk framework en hun beveiliging. |
| Insecure Deserialization | * Zet stricte types tijdens een deserialization voordat een object gecreeerd wordt. Het is mogelijk om hier langs te komen dus leg hier niet alle vertrouwen in. * Log deserializations die falen, zoals waar de incoming type is niet gelijk aan het verwachte type. * Zet restricties en monitor inkomende en uitkomende connecties van containers of servers die deserializen. |
| Insufficient Logging & Monitoring | * Alle authorization requests die falen worden met genoeg gebruiker context gelogt, zodat de verdachte accounts eruit gefilterd kunnen worden. * Zorg dat de logs in een format is dat makkelijk kan worden opgeslagen in log management oplossingen. |

## Hoofdvraag

### Hoe kan het project GamingLads voldoende beveiligd worden?

Het project kan voldoende beveiligd worden als er wordt voldaan aan de punten die naar voren kwamen in deelvraag drie. Dit zijn Broken Authentication, Sensitive Data Exposure, Cross-Site Scripting XSS, Insecure Deserialization en Insufficient Logging & Monitoring. In het algemeen betekent dit: meer logging, meer restricities, en meer checks op requesten en wachtwoorden. Dat is op dit moment nog vrij matig.

## Bronnen

<https://owasp.org/www-project-top-ten/>

<https://owasp.org/www-project-top-ten/2017/A1_2017-Injection.html>

<https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication.html>

<https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure.html>

<https://owasp.org/www-project-top-ten/2017/A4_2017-XML_External_Entities_(XXE).html>

<https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control.html>

<https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration.html>

<https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS).html>

<https://owasp.org/www-project-top-ten/2017/A8_2017-Insecure_Deserialization.html>

<https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities.html>

https://owasp.org/www-project-top-ten/2017/A10\_2017-Insufficient\_Logging%2526Monitoring.html

Testrapport GamingLads