OWASP report

28-04-2021

DOOR: VINCENT DARWINKEL

INLEIDING

In dit document staat de OWASP top 10 beschreven met betrekking tot het Eindhovense vriendjes (EHV) project. In het OWASP top 10 rapport staan de meest voorkomende beveiligingsproblemen beschreven die kunnen voorkomen in software projecten. In dit document heb ik een OWASP top 10 onderzoek uitgevoerd op de EHV applicaties.

INHOUDSOPGAVE

OWASP TOP 10

1. INJECTION

Het meest voorkomende probleem is injection. Dit kan SQL, NoSQL, OS, LDAP, etc. zijn.

Dit houdt in dat data van bijvoorbeeld een inputveld rechtstreeks in het document wordt gebruikt, zonder dat dit gefilterd wordt. Hierdoor kunnen commando's worden uitgevoerd in de front-end en kan de achterliggende server worden aangestuurd.

Is het risico acceptabel voor de applicatie:

Nee het risico is niet acceptabel. Dit omdat er gevoelige data in de database zou kunnen staan, dit mag niet zomaar toegankelijk zijn. Injecties zijn voor mij niet acceptabel.

Hoe heeft het risico de implementatie beïnvloed?

Door gebruik van een object Relational Mapping (ORM) wordt een groot deel van SQL injectie afgevangen. SQL injectie is mogelijk in een ORM als er gebruik wordt gemaakt van raw SQL queries. Ikzelf maak geen gebruik van Raw SQL queries.

Conclusie

Door het gebruik van Entity framework core wordt SQL injectie automatisch afgehandeld. Zolang ik geen gebruik ga maken van raw SQL queries zal er geen risico zijn voor SQL injectie.

2. BROKEN AUTHENTICATION

Het tweede probleem is niet goed geïmplementeerde authenticatie. Hierdoor is het soms mogelijk om wachtwoorden, sleutels of sessie tokens te bemachtigen of misbruiken. Hierdoor is het mogelijk om als iemand anders voor te doen.

Is het risico acceptabel voor de applicatie:

Nee het risico is niet acceptabel. Dit omdat er een reden is dat er gebruikersrollen in de applicatie aanwezig zijn die verschillende machtigingen hebben in de applicatie. Als dit misbruikt kan worden zou iemand wijzigingen kunnen aanbrengen die niet mogen.

Hoe heeft het risico de implementatie beïnvloed?

- Ik ga hier actief op testen en tijdens het programmeren rekening mee houden (Security by design)
- In de unit testen wordt ook gekeken naar de beveiliging van de methodes
- Door middel van integratietesten wordt er ook naar beveiligingsproblemen gekeken over heel de flow van de applicatie
- In de CI pipeline wordt gekeken naar beveiligingsproblemen door middel van Snyk
- In de CI pipeline wordt gekeken of er geen gevoelige gegevens aanwezig zijn in de code
- De extensie SonarLint voor Visual Studio kijkt of ik geen kwetsbaarheden in mijn code heb zitten die kunnen leiden tot beveiligingsproblemen. Dit is echter wel beperkt in wat het kan detecteren, SonarLint kan bijvoorbeeld niet zijn of mijn implementatie wel veilig is maar kijkt meer naar buffer overflow attacks etc.

Conclusie

Geen enkele applicatie is 100% veilig. Ook mijn applicatie zal niet 100% veilig zijn. Ik probeer zo dicht mogelijk naar de 100% te komen door gebruik te maken van bewezen frameworks en door best practices toe te passen. Verder maak ik gebruik van geautomatiseerde tooling om te kijken naar problemen in de applicatie.

3. SENSITIVE DATA EXPOSURE

Het derde probleem is het niet goed beveiligen van data. Dit kan bijvoorbeeld het niet versleuteld opslaan van gevoelige gegevens betekenen of geen check doen of iemand wel bij data mag komen.

Is het risico acceptabel voor de applicatie:

Nee het risico is niet acceptabel. Dit omdat er gevoelige gegevens kunnen worden opgeslagen in de applicatie. Deze gegevens zullen goed beveiligd moeten worden.

Hoe heeft het risico de implementatie beïnvloed?

- Door middel van integratietesten wordt er gekeken naar beveiligingsproblemen over heel de flow van de applicaties
- Door middel van unit testen kijk ik of methodes wel veilig omgaan met malafide input
- Door gebruik te maken van GitGuardian wordt er gekeken of er geen credentials in de code aanwezig zijn op Github

Conclusie

Wachtwoorden worden gehased met Argon2 opgeslagen in de database. Bepaalde zaken als email adressen en andere gegevens zouden nog versleuteld opgeslagen kunnen worden hier moet ik nog naar kijken voordat de applicatie in productie gaat.

4. XML EXTERNAL ENTITIES

Het vierde risico is XXE. Dit houdt in dat een gebruiker XML of kwaadaardige XML in een document, kwetsbare code, dependencies of integraties kan verwerken. Deze code kan worden gebruikt om bijvoorbeeld data te verkrijgen, verzoeken vanuit de server te sturen, dos aanval en nog veel meer aanvallen uit te voeren.

Is het risico acceptabel voor de applicatie:
Nee het risico is niet acceptabel. Echter ga ik geen gebruik maken van XML in mijn applicatie. Hierdoor hoef ik er in principe geen rekening mee te houden, maar zal ik het wel in mijn achterhoofd houden.

Hoe heeft het risico de implementatie beïnvloed?
N.v.t.

Conclusie
N.v.t.

5. BROKEN ACCESS CONTROL

het vijfde risico is niet goed werkende toegangscontrole. Dit betekent dat wat een gebruiker niet mag doen niet goed wordt gehandhaafd in de applicatie. Aanvallers kunnen misbruik maken van deze kwetsbaarheden. Hierdoor kunnen ze bijvoorbeeld toegang krijgen tot een functionaliteit waar ze niet bij mogen komen.

Is het risico acceptabel voor de applicatie:

Nee het risico is niet acceptabel. Gebruikers zouden dan bijvoorbeeld planningen kunnen aanpassen wat niet de bedoeling is.

Hoe heeft het risico de implementatie beïnvloed?

- Door middel van integratietesten wordt er gekeken naar beveiligingsproblemen over heel de flow van de applicaties
- Door middel van unit testen kijk ik of de autorisatie van methodes wel op order is

Conclusie

Dankzij testen kan er al veel afgevangen worden. Verder is handmatig testen ook belangrijk en met beveiliging in gedachten programmeren.

6. SECURITY MISCONFIGURATION

Het zesde risico is verkeerde configuratie. Een voorbeeld is gebruik maken van http i.p.v. https en data die toegankelijk is zonder authenticatie.

Is het risico acceptabel voor de applicatie:
Nee het risico is niet acceptabel. Tijdens mijn opleiding MBO ICT beheer heb ik mij bezig gehouden met het beveiligd instellen van server. Dit ga ik hanteren bij de implementatie van de applicatie.

Hoe heeft het risico de implementatie beïnvloed?
Bij de configuratie van de servers ga ik rekening houden met het beveiligd instellen hiervan. Verder ga ik onderzoeken uitvoeren voor het veilig gebruik maken van Kubernetes en Docker

Conclusie
Tijdens het invullen van het configuratie bestand moet goed worden opgelet dat de juiste gegevens worden ingevuld. Verder moet degene die de applicaties implementeert ervoor zorgen dat de applicaties via HTTPS enkel zijn te benaderen. Daarnaast moet ook gekeken worden naar de configuratie van Kubernetes of dit wel veilig is.

7. CROSS SITE SCRIPTING

Het zevende risico is XSS. Dit houdt in dat bijvoorbeeld bij een gastboek Javascript code kan worden geïmplementeerd, wordt opgeslagen en wordt uitgevoerd als een andere gebruiker het gastenboek bezoekt. De Javascript code kan bijvoorbeeld de cookies van de gebruiker stelen en doorsturen naar een server.

Is het risico acceptabel voor de applicatie:
Nee het risico is niet acceptabel. Hier ga ik op filteren in de back-end.

Hoe heeft het risico de implementatie beïnvloed?
Ik ga een middleware implementeren die cross site scripting detecteert in datamodellen. Op basis hiervan is het niet mogelijk om in bijvoorbeeld een database XXS code op te slaan.

Conclusie
React zelf beschermd al gedeeltelijk tegen XSS. Verder wordt in de backend gekeken naar XSS aanvallen en worden deze verzoeken geweigerd.

8. INSECURE DESERIALIZATION

Het achtste risico is onbeveiligde zuivering. Dit houdt in dat code uitgevoerd kan worden die niet uitgevoerd mag worden. Voorbeelden zijn injection aanvallen en replay aanvallen.

Is het risico acceptabel voor de applicatie:
Nee het risico is niet acceptabel. Hier ga ik op filteren in de back-end en front-end.

Hoe heeft het risico de implementatie beïnvloed?
Ik ga hier actief op testen en tijdens het programmeren rekening mee houden. Ook ga ik bij de configuratie van de server controleren op de rechten.

Conclusie
Replay attacks moet ikzelf onderzoek naar gaan doen omdat dit nog niet behandeld is. Hier zit dus nog een potentieel beveiligingsprobleem in.

9. USING COMPONENTS WITH KNOWN VULNERABILITIES

Het negende risico is componenten gebruiken waarvan bekend is dat er kwetsbaarheden inzitten.

Is het risico acceptabel voor de applicatie:

Ja en nee het risico is niet acceptabel, maar als er een work around voor is die het probleem oplost zou ik er minder problemen mee hebben. Zolang de work around goed werkt.

Hoe heeft het risico de implementatie beïnvloed?
Door gebruik te maken van Snyk in de CI pipeline krijg ik een melding als er verouderde libraries met beveiligingsproblemen worden gebruikt. Snyk kan zelfs pull requesten aanmaken om de problemen op te lossen.

Conclusie
Zowel Snyk en NPM (NPM geldt voor frontend, Snyk werkt voor front en backend) testen op verouderde libraries. Hierdoor heb ik een goed overzicht van verouderde en onveilige libraries in het project.

10. INSUFFICIENT LOGGING & MONITORING

Het tiende risico is het niet goed loggen van acties en het monitoren hierop. Hierdoor kan een aanvaller mogelijk onopgemerkt zijn gang gaan.

Is het risico acceptabel voor de applicatie:
Nee het risico is niet acceptabel en ik zal hier rekening mee houden.

Hoe heeft het risico de implementatie beïnvloed?
Ik ga monitoring in de applicatie inbouwen die kan detecteren of iemand een poging doet tot een aanval. Zodra dit gedetecteerd wordt er een mail verstuurd naar mij. Verder kijkt de logging of er eventuele bugs in de code zitten.

Conclusie

De monitoring moet nog worden geïmplementeerd.