

Mohamed El Kawakibi

Proof of Concept

Webapplicatie scanner

Inhoud

[4](#_Toc495611396)

[1 Inleiding 5](#_Toc495611397)

[1.1 aanleiding 5](#_Toc495611398)

[1.2 Probleemstelling 5](#_Toc495611399)

[Gratis scan 5](#_Toc495611400)

[1.3 Scope 6](#_Toc495611401)

[1.4 Onderzoek opzet 7](#_Toc495611402)

[1.5 Project context 8](#_Toc495611403)

[2. Business Context 10](#_Toc495611404)

[2.1 Stakeholders en Business doelen 10](#_Toc495611405)

[2.1.1 Stakeholders 10](#_Toc495611406)

[2.1.2 Business Doelen 12](#_Toc495611407)

[2.2 Doelgroep 14](#_Toc495611408)

[2.2.1 Doelgroep onderzoek 14](#_Toc495611409)

[3 Onderzoek 18](#_Toc495611410)

[3.1 Inleiding 18](#_Toc495611411)

[3.2 Web security en de kwetsbaarheden 19](#_Toc495611412)

[3.2.1 OWASP top 10 lijst 20](#_Toc495611413)

[3.2.2 OWASP Selectie 21](#_Toc495611414)

[A1 injectie 21](#_Toc495611415)

[A3 XSS 22](#_Toc495611416)

[A5 Security Misconfiguration 23](#_Toc495611417)

[3.2.3 WASC 24](#_Toc495611418)

[3.4 Webapplicatie scanner 27](#_Toc495611419)

[3.4.1 Inleiding 27](#_Toc495611420)

[3.4.4 Acunetix 29](#_Toc495611421)

[3.4.5 Zap Zed Attack Proxy 30](#_Toc495611422)

[3.4.7 Conclusie 32](#_Toc495611423)

[3.5 Technologie 34](#_Toc495611424)

[3.5.1 Web Application Framework 34](#_Toc495611425)

[3.5.2 Database 37](#_Toc495611426)

[4 Concept ontwikkeling 39](#_Toc495611427)

[4.1 Concept van het system (SecurityReport) 41](#_Toc495611428)

[4.2 CMS extensies 47](#_Toc495611429)

[4.3 API 50](#_Toc495611430)

[4.4 Webapplicatie scanner 51](#_Toc495611431)

[4.4.2 Crawler 54](#_Toc495611432)

[4.4.3 Scanner 59](#_Toc495611433)

[4.4.4 Rapporteren 68](#_Toc495611434)

[5 Eindproduct 72](#_Toc495611435)

[Bronnenlijst 73](#_Toc495611436)

[Bijlage A 74](#_Toc495611437)

[Glossary 74](#_Toc495611438)

# 

**INLEIDING**

# 1 Inleiding

## 1.1 aanleiding

Het is algemeen bekent dat websites/webapplicaties kwetsbaarheden hebben waar hackers gebruik van maken door deze te exploiteren. Deze kwetsbaarheden worden veroorzaakt door onveilige geschreven code en misconfiguratie aan de server kant van de applicatie. Een gros van de webapplicatie die nu online staan laten input van de gebruikers toe, dit om de gebruiker te kunnen laten communiceren met de server om bijvoorbeeld data op te vragen vanuit de database. De hackers maken van deze mogelijkheid gebruik van door code te injecteren die de server zonder het te valideren uitvoert. De codetalen die hiervoor gebruikt kunnen worden zijn SQL een database taal en browser script talen (Javascript, VB).

S5 specialiseert zich in het ontwikkelen van website, in het specifiek webshops. Hiervoor gebruiken zij de Content Management Systemen (CMS) platformen WordPress en Magento, twee CMS-platformen die zoals bijna elke webapplicaties security kwetsbaarheden hebben. Hiervoor heeft S5 aanleiding genoeg om dit probleem te onderzoeken, omdat er altijd een kans is dat één van hun webshops wordt aangevallen door een cybercrimineel die uit zijn op het aanrichten van schade. Dit gaf S5 de motivatie om hiervoor maatregelingen te nemen. Ik heb de opdracht gekregen om het probleem te onderzoek en een gepaste oplossing als aanbeveling te geven in de vorm van een webapplicatie dat webshops op kwetsbaarheden scant. De Magento CMS valt buiten de scope van deze opdracht en zal niet verwerkt worden in de oplevering van de webapplicatie scanner.

## 1.2 Probleemstelling

In recente onderzoek is gebleken dat CMS systemen als WordPress een veelvoorkomende doelwit is voor hackers. Er zijn hier twee reden voor, 27% van alle website, wat een ongeveer 1 miljard is, wordt aangedreven door het WordPress CMS platform. Dit betekent dat wanneer er een nieuwe kwetsbaarheid gevonden wordt dat, er meer dan 200 miljoen websites in gevaar lopen voor cyberaanvallen. De andere reden is dat WordPress extensies gebruikt, deze worden ontwikkeld door derde partijen, sterker nog iedereen met WordPress en PHP kennis zou een extensies kunnen maken en publiceren op de Extensie markt van WordPress.

Het zou ideaal zijn als webshop eigenaren in het specifiek de klanten van S5 inzicht zouden krijgen op de kwetsbaarheden van hun webshop. Informatie als onveilige inputvelden en misconfiguraties van de webserver/webshop kunnen de webshop eigenaren inzicht bieden in welke riciso’s hun webapplicaties lopen. Het eerste probleem waar al een oplossing voor is bedacht is: hoe kan S5 ervoor zorgen dat de webshop eigenaren inzichtelijke informatie kunnen verkrijgen over de beveiligingsstaat van hun webshop. De oplossing voor deze vraagstelling is ook tevens de aanleiding van mijn opdracht het onderzoeken en bouwen van een webapplicatie scanner, wat zal moeten dienen als een Proof of Concept.

### Gratis scan

De klanten van S5 zijn over het algemeen MKB-webshop ondernemers en doen daarom hun best om een goede klantenservice te bieden. Maar wat als een hacker er vandoor gaat met de klantgegevens heeft de webshop dan nog recht om klantenservice hoog in het vaandel te zetten? Wat kan hiertegen gedaan worden. In het algemeen kan een webshop eigenaar securitymaatregelingen nemen om de webshop beter te beveiligen. Hiervoor kan de webshop eigenaar contact opnemen met een cybersecurity bedrijf dat consultancy aanbiedt, dit kan hoog in de kosten lopen. MKB-webshop hebben het al moeilijk genoeg (mkbservicedesk.nl, Problemen bij financieren van ondernemers). De reden hiervoor is omdat reuzen zoals Wehkamp.com diensten kunnen aanbieden zoals gratis verzenden en same day delivery, die voor MKB-bedrijven in financiële en logistieke opzicht lastig te realiseren zijn. Een web security scan kan in de duizenden euro’s lopen wat menig webshop eigenaren als een overbodige kostenpost zien en ik geef ze tot een bepaalde hoogte wel gelijk maar als een hacker jouw webshop in het vizier heeft dan is de vraag nog hoe veel schade er zal worden toe gericht.

De website sectoolmarket.com, een vergelijking website voor webapplicatie scanners, heeft in 2016 een lijst gepubliceerd met informatie over de prijzen en features van de meeste gebruikte webapplicatie scanners. Ik heb naar de 15 commerciële producten gekeken en daar de drie duurste uitgekozen. Ik heb tevens ook naar de open-source producten gekeken hiervan waren er 49, een zeer grote aanbod. Het probleem hiermee is dat zij zeer beperkt zijn in hun scan opties en niet specifiek zijn gericht op de doelgroep van S5. Ook is het zo dat het zeer lastig om deze open-source producten te integreren in een test omgeving omdat er niet genoeg aandacht is besteed aan het integratieproces van het product oftewel deployment.

Bedrijven als IBM, Acunetix, Netsparker bieden de commerciële producten aan en hebben daarmee ook de hoogste kostenplaatje als het gaat om diensten/producten zoals consultancy en enterprise applicaties maar in vergelijking met de open-source applicaties bieden zij meer features aan en zijn de applicaties veel accurater als het neer komt op het scannen naar kwetsbaarheden.

|  |  |  |  |
| --- | --- | --- | --- |
| Commerciële producten | | | |
| Naam | Features | Prijs consultancy per jaar | Prijs applicatie |
| IBM | SQLi, XSS, file inclusion | 17700$ | - |
| Acunetix | SQLi, XSS, file inclusion | 3500$ | 2495$ |
| Netsparker | SQLi, XSS, file inclusion | 3960$ | 3960$ |

*Tabel 1.1.1 – commerciële producten*

In tabel 1.1.1 zie je hoe hoog de kosten kunnen zijn, open-source applicatie zijn daarentegen gratis te gebruiken maar vergen veel technische kennis. Nu zou een webshop eigenaar een opensource applicatie kunnen aanschaffen maar, dan is de vraag nog of hij/zij instaat is om een succesvolle implementatie in de praktijk te brengen. Wat er hoogstwaarschijnlijk gaat gebeuren is dat er een expert wordt ingehuurd om alles op te zetten, ook dit kan hoge kosten opleveren. Een gratis webapplicatie scanner dat makkelijk te gebruiken is voor een bestaande klant van S5 ideaal en zou tevens de kostenpost van consultancy en dure licentie omzeilen.

## 1.3 Scope

Omdat ik mij zal bezighouden met het ontwikkelen van een Minimal Viable Product (MVP) zal ik het project moeten afbakenen. Een Webapplicatie scanner kan zeer uitgebreide features bevatten zoals u in hoofdstuk 3.4 zal merken. Voor de scope van dit project heb ik een aantal onderwerpen bestudeerd zoals de business context van het project, wat is het doel van dit project met betrekking tot het zakelijke deel ervan. De kwetsbaarheden die in een webapplicatie kunnen voorkomen en de verschillende organisaties die zich bezighouden met het identificeren van deze kwetsbaarheden. Er zijn heel wat kwetsbaarheden die een website kan hebben, en er komen er nog elke week nieuwe bij die gelabeld worden als zero-day kwetsbaarheden. De webapplicatie scanner, uit welke fundamentele componenten bestaat het. De programmeer frameworks en de databases die verschillende organisaties beschikbaar hebben gesteld als open-source projecten. De Customer Journey, welke stappen moet de klant nemen om succesvol een scan aan te vragen. Al deze onderwerpen heb ik binnen een vastgestelde scope gedefinieerd om het project haalbaar te maken en om het te laten voldoen aan de eisen van een Minimal Viable Product.

|  |
| --- |
| Scope van de scriptie |
| Onderzoek |
| Concept |
| Software Architectuur |
| Realisatie |

|  |
| --- |
| Scope van het softwaresysteem |
| WordPress extensie |
| Webapplicatie scanner |
| Admin panel |
| API |

## 1.4 Onderzoek opzet

**Business Context**

Ik zal beschrijven wat het doel is van dit project, wie de stakeholders zijn en wat de doelgroep is.

**Kwetsbaarheden**

Ik zal onderzoeken wat de meest voorkomende kwetsbaarheden zijn van een website en welke het best passen bij de doelgroep.

**Rapporten**

Dit onderzoek dient om meer context te geven aan het probleem voor de lezer en voor mij. Zodat we weten wat de huidige situatie is van web security. Voor meer inzicht in het probleem zal ik een aantal rapporten van grote bedrijven behandelen. Dit dient voor extra context die de lezer kan gebruiker om een beter inzicht te krijgen van het probleem.

**Webapplicatie scanners (DAST)**

Om inzicht te krijgen in het proces van een scan zal ik een aantal bestaande scan applicatie testen om de resultaten te kunnen analyseren. Dit dient om mij meer informatie te geven over de werking van een webapplicatie scanner en wat ik nodig heb om er één zelf te bouwen.

**Webapplicatie Framework & Database**

**Disclosure**

Ik heb tijdens het solliciteren ook specifiek gezocht naar stages die PHP opdrachten aanbieden. S5 bestaat uit twee afdelingen PHP en C#, ik zit aan de PHP kan, wat betekent dat mijn directe collega’s die mij ondersteunen in het ontwikkelen van het softwareapplicatie gespecialiseerd zijn in de PHP programmeertaal. Om deze twee redenen is het vrij logisch dat ik voor een PHP Framework heb gekozen. Dit is voor het review van code en het oplossen van bug ideaal. PHP biedt heel wat Web Application Frameworks aan waaronder Laravel, Slim, Symfony, Zend, Phalcon, Yii. Maar welke is het best geschikt voor het project?

In dit onderzoek zal ik de tools die voor ons beschikbaar zijn gesteld binnen de scope onderzoeken. Dit onderzoek is gericht op het maken van de juiste keuze wat betreft de Webapplicatie Framework en Database tool die ik ga gebruiken voor het ontwikkelen van de applicatie.

Code voorbeelden die in dit documenten behandeld worden, zijn in de taal PHP geschreven.

**Bijlage A en B**

In Bijlage A vindt u theorie gerelateerde onderwerpen en in Bijlage B uitvergrote systeem diagrammen.

## 1.5 Project context

**Doelstelling**

Het doel is om een Proof Of Concept te ontwikkelen voor een gratis Webapplicatie scanner die bestaande klanten van S5 kunnen gebruiken om hun webshop op kwetsbaarheden te kunnen laten scannen. Na de scan zal er een rapport gemaakt worden met de geïdentificeerde kwetsbaarheden en advies voor het oplossen van de securityproblemen.

**Opdracht**

De opdracht is het onderzoeken en ontwikkelen van een systeem dat WordPress website scant op kwetsbaarheden. De opdracht zal bestaan uit drie sub opdrachten: onderzoeken van het probleem, het ontwikkelen van een concept en het ontwikkelen van een prototype.

**Hoofdvraag**

Hoe kan ik een MVP van een geautomatiseerde webapplicatie scanner ontwikkelen, die kwetsbaarheden identificeert en op basis daarvan een rapport genereert, waar de klanten van S5 gratis gebruik van kunnen maken?

**Deelvragen**

* Voor wie is de Webapplicatie scanner bedoelt?
* Wat is een Webapplicatie scanner?
* Wat zijn de tools en technieken die ik kan gebruiken bij het ontwikkelen van een Webapplicatie scanner?
* Hoe ziet het concept eruit van de Webapplicatie scanner?
* Wat is het ontwikkelproces voor het realiseren van de Webapplicatie scanner?



**BUSINESS CONTEXT**

**Deelvraag**

**Voor wie is de Webapplicatie scanner bedoelt?**

# 2. Business Context

**Inleiding**

In dit deel zal ik behandelen wat het business context is van het project. Er zal onderzoek gedaan worden naar de business doelen van het project, wie de stakeholders zijn en wat de doelgroep is.

|  |
| --- |
| Business doelen |
| Stakeholders |
| Doelgroep |

Tabel 2.1

## 2.1 Stakeholders en Business doelen

Het is voor Stakeholders van belang dat zij weten wat het nut is van een systeem. Als het niet duidelijk is waarom een systeem wordt gebouwd dan is er een grote kans dat het doeleinde ook niet volledig wordt bereikt. De vragen die gesteld moeten worden voor het definiëren van het nut en doeleindes van een systeem zijn:

* Wat zijn de vastgestelde doelen voor het nieuwe Geautomatiseerde Webapplicatie scanner Systeem?
* Wie zijn de stakeholders?
* Wie is de doelgroep?

### 2.1.1 Stakeholders

Het identificeren van stakeholders is meestal het eerste stap in het schrijven van een Business Georiënteerde Document. Stakeholders zij de mensen, groepen, of instituten die beïnvloed kunnen worden door de uitkomst van bijvoorbeeld een project.

Om de stakeholders te identificeren gebruik ik een brainstorm techniek, het resultaat ervan is geïllustreerd in figuur 2.1.

Figure 2.1

Het identificeren van stakeholders behoort tot één van de primaire taken voor een project. De stakeholders die niet zijn geïdentificeerd kunnen ook niet betrokken worden bij het project. De stakeholders bieden je ook meer inzicht op het probleem, omdat je het dan vanuit hen hoek naar het probleem kunt kijken.

De geïdentificeerde stakeholders bestaan uit twee categorieën de primaire (wit) en secondaire (Grijs) stakeholders.

* Primaire stakeholders worden direct beïnvloed (positief of negatief).
* Secondaire stakeholders worden mogelijk beïnvloed door de primaire stakeholders (positief of negatief)

|  |  |  |
| --- | --- | --- |
| ID | Stakeholders | Categorie |
| 1 | S5 | Primaire |
| 2 | Klanten van S5 | Primaire |
| 3 | Ontwikkelaars | Primaire |
| 4 | Administrator | Primaire |
| 5 | Concurrentie | Secondaire |
| 6 | GitHub | Secondaire |
| 7 | Bijdragers (ontwikkelaars) | Secondaire |
| 8 | Programmeer Libraries | Secondaire |

Tabel 2.2

**Primaire stakeholders**

De primaire stakeholders zijn de meest van belang voor het project, deze stakeholders hebben direct invloed op de ontwikkeling van het project. Ieder stakeholder heeft zijn rol binnen het project en het is de bedoeling om met alle primaire stakeholders het project tot een succes te maken.

**Rollen**

* S5 – De opdrachtgevers van het project en product owners. Zij zijn deels verantwoordelijk voor het goedkeuren van userpoints (Glossary, #)
* Ontwikkelaars – Zij zijn verantwoordelijk voor het ontwikkelen van het softwaresysteem (Webapplicatie scanner)
* Klanten S5 – Zij zijn het die de eindgebruikers zijn van het softwaresysteem, hun belangen staan vooraan in het project
* Administrator – Zij beheren het softwaresysteem na dat het live is gegaan. Voorbeeld het beheren van scans en rapporten

### 2.1.2 Business Doelen

Wat zijn de Business Doelen?

Wat wilt S5 bereiken met dit systeem in het kader van Business?

S5 wilt een systeem bouwen met als hoofddoel een gratis geautomatiseerde website applicatie scanner aan te bieden. S5 is een onderneming en wilt graag dat het blijft groeien en uitbreiden. De business goals zijn zeer algemeen en gangbaar bij veel bedrijven, welke bedrijf wilt nou niet zijn services uitbreiden en de klantenrelaties versterken en consultancy aanbieden als verdienmodel. Dat het algemeen en cliché is betekent niet dat deze business goals geen impact kunnen hebben op de groei van een bedrijf. Juist voor een onderneming als S5 zijn dit de doelen waar zij naar moeten streven. Als het bedrijf weet wat het wilt bereiken dan is de volgende vraag: Hoe gaan we deze business goals bereiken?

* Services aanbod uitbreiden
* Het servicekwaliteit verbeteren

|  |  |
| --- | --- |
| Doel Categorie | Doel omschrijving |
| Service aanbod uitbreiden | Het systeem breid de service uit door   * Een gratis geautomatiseerde webapplicatie scanner aan te bieden * Door een CMS extensie aan te bieden * Door een consultancy service aan te bieden |
| Het servicekwaliteit verbeteren | Het systeem verbeterd het servicekwaliteit door   * Een rapport genereren op basis van de geïdentificeerde kwetsbaarheden * Website verbeteren met de aanbevelingen van het security rapport * Security experts in dienst nemen om de security issues op te lossen en om consultancy te geven |

Tabel 2.3

**Business doel Scenario**

De business doel scenario beschrijft elke doel meer in detail door gebruik te maken van de zes attributen waar een business doel uit bestaat. Dit zorgt ervoor dat business doelen concreet en uitvoerbaar worden.

De zes attributen zijn:

|  |  |
| --- | --- |
| Doel Subject (DS) | De stakeholder die eigenaar is van het doel |
| Doel Object (DO) | De entiteit van het doel |
| Omgeving (O) | De context van het doel |
| Doel (D) | Het doel zelf |
| Doel Maatregel (DM) | Het succes criteria van het doel |
| Achtergrond Informatie (Ai) | Achtergrond informatie over het doel |

Tabel 2.4

|  |  |  |
| --- | --- | --- |
| **Doel Categorie** | **Informele business doel statement** | **Business Doel Scenario** |
| Service aanbod uitbreiden | We moeten de service aanbod uitbreiden door een geautomatiseerde Webapplicatie scanner te introduceren. | |  | | --- | | DS: S5 | | DO: Webapplicatie scanner | | O: Service aanbod uitbreiden | | D: Een nieuwe service aanbieden | | DM: Klanten kunnen een gratis geautomatiseerde scan uitvoeren | | AI: S5 wilt de klanten een tool aanbieden om hun websites te scannen op security | |
| Het servicekwaliteit verbeteren | We moeten naast het ontwikkelen van webshop ook meer de focus leggen op security door de klant meer inzicht te bieden op de security staat van hun webshop. | |  | | --- | | DS: S5 | | DO: Security Rapport | | O: Kwaliteit service verbeteren | | D: Security Rapporten genereren op basis van de gevonden kwetsbaarheden | | DM: Klanten inzicht bieden in de security van hen webshops | | AI: S5 wilt de klanten meer overzicht bieden op de security van hen webshops | |

Tabel 2.5

**Business doelen verfijnen**

Het doel van het verfijnen van de business doelen is om de relevantie ervan te vinden met betrekking tot het softwaresysteem dat onderontwikkeling is. Verfijnde business doelen worden operationele objectieven en als je daaruit de relevante operationele objectieven haalt dan hou je engineering objectieven over (Software and Systeem Architecture In Action, H 2.4 Refining Business Goals). In het geval van het project resulteren de twee business doelen in relevante operationele objectieven vanwege hun betrekking op het systeem.

|  |  |
| --- | --- |
| Business Doelen | Verfijnde doelen (Engineering Objectieven) |
| We moeten de service aanbod uitbreiden door een geautomatiseerde Webapplicatie scanner te introduceren. | |  | | --- | | Een Client applicatie voor de klant wat dient als Endpoint (Glossery, 13) | | Een communicatiemiddel tussen de Client en de Server | | De Wep Applicatie Scanner, Crawler, Scanner, Rapporteren | |
| We moeten naast het ontwikkelen van webshop ook meer de focus leggen op security door de klant meer inzicht te bieden op de security staat van hun webshop. | |  | | --- | | Het genereren van een rapport op basis van de gevonden kwetsbaarheden | | Het verzenden van het rapport via email | |

Tabel 2.6

**Engineering objectieven, kwaliteit attributen en prioriteiten**

Engineering objectieven kunnen gebruikt worden om de systeemeisen van de systeemarchitectuur te definiëren. Elke engineering objectief heeft een corresponderende kwaliteit attribuut, deze vormen de basis voor het definiëren van de systeemeisen. Als voorbeeld kunnen kijken naar de business doel, het behouden van de markt reputatie. Dit business doel correspondeert met de engineering objectief: het maken van producten dat zeer Usable en Reliable is. (Het kwaliteit attributen kun je vinden in Bijlage A).

|  |  |  |  |
| --- | --- | --- | --- |
| Business Doelen | Verfijnde doelen (Engineering Objectieven) | Kwaliteit Attribuut | Prioriteit |
| We moeten de service aanbod uitbreiden door een geautomatiseerde Webapplicatie scanner te introduceren. | |  | | --- | | Een Client applicatie voor de klant wat dient als Endpoint (Glossery, 13) | | Een communicatiemiddel tussen de Client en de Server | | De Webapplicatie scanner, Crawler, Scanner, Rapporteren | | |  | | --- | | Usability | | Interoperability | | Performance | | |  | | --- | | M | | M | | H | |
| We moeten naast het ontwikkelen van webshop ook meer de focus leggen op security door de klant meer inzicht te bieden op de security staat van hun webshop. | |  | | --- | | Het genereren van een rapport op basis van de gevonden kwetsbaarheden | | Het verzenden van het rapport via email | | |  | | --- | | Modifiability | | Availibility | | |  | | --- | | H | | M | |

Tabel 2.7

## 2.2 Doelgroep

S5 ontwikkelt en onderhoudt webshops voor MKB-bedrijven. De webshops worden over het algemeen ontwikkeld in de CMS Wordpress. WordPress heeft zijn CMS in de taal PHP ontwikkeld en heeft daarvoor een eigen framework voor gebruikt.

|  |  |
| --- | --- |
| Doelgroep | Categorie |
| Wordpres | Webshop |

Tabel 2.8

### 2.2.1 Doelgroep onderzoek

Webshops die ontwikkeld zijn in Wordpress zijn kwetsbaar voor cyberaanvallen. 27% procent van alle websites draaien op WordPress en het heeft 60% van het marktaandeel als het gaat om CMSen (Torquemag, 2xxx). Dit betekent, in theorie, dat als het CMS een kwetsbaarheid heeft, dan zijn 27% van alle websites kwetsbaar voor cyberaanvallen. Ithemes vermeld in het artikel “is WordPress Really Secure?” dat de WordPress CMS kampt met securityproblemen. De securityproblemen komen van drie componenten af: WordPress Core, WordPress themes, WordPress plugins. De grootste deel van de security kwetsbaarheden komt niet af van de Kern van WordPress maar van de WordPress plugins. De WordPress plugins rekenen voor 52% van alle kwetsbaarheden.







**ONDERZOEK**

**Deelvraag**

**Wat is een webapplicatie scanner?**

# 3 Onderzoek

## 3.1 Inleiding

Dit hoofdstuk zal gaan over mijn onderzoek die de basis legt voor de Webapplicatie scanner. In dit hoofdstuk heb ik verschillende onderwerpen rondom web security onderzocht. Het onderzoek is grotendeels gedaan vanaf het bureau, wat ook wel betekent staat als bureauonderzoek of deskonderzoek. Deze soort onderzoek kan verdeeld worden in twee categorieën: primaire data en secundaire data. De categorie primaire data heeft in de context van mijn onderzoek betrekking op software testing. Het is primaire omdat ik zelf de software tests en hierdoor zelf de informatie produceer. Bij een secundair bureauonderzoek heb ik gebruik gemaakt van bestaande informatie dat door andere is geproduceerd. Voorbeelden hiervan zijn online artikelen, literatuur, informatieve video’s.

De onderwerpen die ik in dit hoofdstuk zal behandelen zijn:

Web security en de kwetsbaarheden

Een onderzoek naar de kwetsbaarheden van websites. Hierin zal ik kijken naar de kwetsbaarheden die kunnen voorkomen op een website. Het aanbod van technologieën voor het ontwikkelen van website wordt met het jaar groter en dat geeft de web ontwikkelaars de mogelijkheid om websites op verschillende manieren te bouwen. Maar innovaties in web ontwikkeling brengen ook securitylekken met zich mee waar cybercriminelen misbruik van kunnen maken.

Wat zijn de kenmerken van bestaande web scan applicaties?

Met de opkomst van webapplicaties kunnen gebruiker een interactieve ervaring beleven op het internet. Dit maakt mogelijkheid dat gebruikers taken kunnen uitvoeren zoals: persoonlijke accounts te maken, informatie op te vragen vanuit een database en digitale transacties te maken. Al deze nieuwe functionaliteiten brengen ook nieuwe gevaren met zich mee, hackers die kwetsbaarheden in deze functionaliteiten vinden hebben de mogelijkheid om het te exploiteren. Dit is hoofdzakelijk waarom er organisaties zijn die zich bezighouden met het ontwikkelen van tools waarmee webapplicaties getest kunnen worden op kwetsbaarheden zodat deze in een vroeg stadium gedetecteerd kunnen worden. De security scanner is een relatief nieuwe begrip, wat pas halverwege de eerste decennia zich heeft gevestigd als een volwaardige kandidaat op de softwaremarkt. Er zijn talloze webapplicatie scanners ontwikkelt over de jaren heen. Ik zal in deze paragraaftwee van deze soort scanners onder de loep nemen en testen.

Welke technologieën zijn geschikt voor de Webapplicatie scanner?

Er zijn vele programmatalen waarmee software geschreven kan worden maar welke is het meest geschikt voor mijn situatie. In dit deel heb ik onderzoek gedaan naar de verschillende technologieën die ik kan gebruiken voor het ontwikkelen van de Webapplicatie scanner. Een webapplicatie bestaat uit subsystemen in het geval van de Webapplicatie scanner zijn deze: CMS extensie, REST API, Scan applicatie en de Database. Elke subsysteem zal met een ander technologie moeten worden ontwikkeld. Ik zal per subsysteem de verschillende technologieën vergelijken om zo een goed onderbouwde keuze te kunnen maken.

Hoe groot is het security probleem?

Wat rapporteren de grote bedrijven over het probleem web security. In dit deel heb ik de analyse en rapportage van drie grote security bedrijven onder de loep genomen om te kijken wat hun onderzoek naar web security over de recente jaren hebben opgeleverd. Web security blijft ieder jaar een interessant onderwerp van onderzoek vanwege de nieuwe ontwikkelingen op softwaregebied die securitybedreigingen met zich mee brengen. Daarom brengen iedere jaar bedrijven als IBM, Symantec en Sucuri rapporten uit om de geïnteresseerde te informeren over de stand van zaken op securitygebied. Bijlage

## 3.2 Web security en de kwetsbaarheden

Web security is een vertakking van Informatiebeveiliging dat zich specifiek bezighoudt met het beveiligen van websites. Web security werd echter pas een probleem met de komst van web 2.0. Bij de voorganger, web 1.0 konden gebruikers voornamelijk alleen informatie te bekijken. Bij web 2.0 kregen gebruikers meer rechten op een website, hierdoor konden gebruikers naast het bekijken van informatie ook informatie creëren en bewerken. Voorbeeld hiervan is het aanmaken van accounts voor een internetdienst. Deze veranderingen brachten niet alleen innovatie met zich mee maar ook veel security issues. Een van deze security issues gebeurt bij het verzenden van informatie naar servers in de vorm van een requests (verzoeken), wat een vorm is van communicatie tussen gebruiker en server. Hackers gebruiken deze communicatiemiddel om websites te comprimeren door kwaadaardige requests te verzenden om bijvoorbeeld een website offline te halen.

Er zijn een aantal non-profit organisaties die zich bezighouden met het onderzoeken, documenteren en ontwikkelen van web security applicaties. Die doen zij door open source projecten te starten waar iedereen met een interesse in web security aan kan meedoen. Er zijn twee wel bekende organisaties, OWASP en WASC. Beide hebben over de jaren heen succesvolle projecten hebben geproduceerd en hebben een actieve community die de projecten onderhouden.

OWASP

De Open Web Application Security Project, opgericht in 2001, is een wereldwijde non-profit organisatie dat zich bezighoudt met het verbeteren van web security. De missie van OWASP vangt één van de essentiële concepten van software security. Hun missie is om software security zichtbaar te maken, zodat individuen en organisaties geïnformeerde keuzes kunnen maken. Iedereen kan een bijdragen met de open-source projecten van OWASP, het heeft een gratis en open software licentie. OWASP heeft over 93 actieve projecten waarvan er een aantal gelabeld worden als volwassen. Relevante projecten die ik zal behandelen in dit document zijn:

|  |  |  |
| --- | --- | --- |
| Classificatie | Type | Project |
| Volwassen | Documentatie | Top 10 |
| Volwassen | Tool | ZAP Proxy |
| Medium | Documentatie | Code Review Project |
| Volwassen | Documentatie | Testing Guide |

Tabel 3.1

WASC

De Web Application Security Consortium is zoals OWASP een organisatie met als missie om web security te verbeteren. Hun hoofdmissie is om een standaard te ontwikkelen voor webapplicatie security. Zij bestaan uit een internationale groep van experts en organisatie vertegenwoordigers die open source en best practice security standaards produceren voor de wereldwijde web. Zij houden zich echter niet bezig met het ontwikkelen van softwaretools. Maar zijn meer gericht op het publiceren van artikelen, onderzoeksrapporten en zoals eerder vermeld standaarden voor web security. WASC classificeert hub projecten niet en het valt mij op dat deze voor een lange tijd niet zijn onderhouden. Relevante projecten die ik zal behandelen document zijn:

|  |  |
| --- | --- |
| Type | Project |
| Documentatie | Web Application Security Scanner Evaluation Criteria |
| Documentaite | WASC Threat Classification |

Table 3.2

### 3.2.1 OWASP top 10 lijst

De OWASP top 10 lijst representeert een overeenstemming over de meest kritische security risico’s voor webapplicaties. OWASP dringt bedrijven aan om dit project te implementeren in het bedrijfsproces om de risico’s op hun webapplicaties te minimaliseren. De lijst is aflopend gesorteerd op de meest voorkomende webapplicatie risico. De 10 security risico’s zijn:

* A1 Injection
* A2 Broken Authentication End Session Management (XSS)
* A3 Cross Site Scripting (XSS)
* A4 Insecure Direct Object References
* A5 Security Misconfiguration
* A6 Sensitive Data Exposure
* A7 Missing Function Level Access Control
* A8 Cross Site Request Forgery Attacks
* A9 Using Components with Known Vulnerabilities Components
* A10 Underprotected API’s

De definities kun je vinden in de bijlage OWASP top tien (Bijlage A).

OWASP heeft in detail beschreven “wat elke risico is, of je webapplicatie kwetsbaar is en hoe het te voorkomen is”? Dit informatie is cruciaal voor het ontwikkelen van een webapplicatie scanner want, het geeft een aan op welke risico’s er minimaal gescand moeten worden. Het is net een criteria lijst voor een webapplicatie scanner. Nu is het een moeilijke taak om binnen de gekregen tijd al deze risico’s te implementeren in de webapplicatie scanner die ik ga ontwikkelen. Omdat ik een Proof of Conecpt en daarom geen uitgebreide webapplicatie scanner hoef te ontwikkelen zal ik maar een selectie van de meest voorkomende risico’s implementeren. Om deze selectie te kunnen maken zal ik kijken naar hoe vaak een risico voorkomt, hoe makkelijk het exploiteerbaar is, wat de impact is op applicatie niveau en business niveau en wat de moeilijkheidsgraad is voor het implementeren van het identificeren van de kwetsbaarheid binnen het softwaresysteem.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Risico | Vaak voorkomend | Exploiteerbaar | Impact software | Impact business | moeilijkheidsgraad |
| A1 | Vaak | Makkelijk | Zeer Hoog | Zeer Hoog | Makkelijk |
| A2 | Vaak | Gemiddeld | Hoog | Hoog | Gemiddeld |
| A3 | Heel vaak | Makkelijk | Hoog | Hoog | Makkelijk |
| A4 | Vaak | Makkelijk | Gemiddeld | Gemiddeld | Moeilijk |
| A5 | Heel Vaak | Makkelijk | Gemiddeld | Hoog | Gemiddeld |
| A6 | Niet Vaak | Moeilijk | Hoog | Hoog | Moeilijk |
| A7 | Niet Vaak | Gemiddeld | Gemiddeld | Gemiddeld | Moeilijk |
| A8 | Vaak | Makkelijk | Gemiddeld | Hoog | Makkelijk |
| A9 | Niet Vaak | Gemiddeld | Gemiddeld | Gemiddeld | Moeilijk |
| A10 | Niet Vaak | Gemiddeld | Gemiddeld | Gemiddeld | Moeilijk |

Tabel 3.3 (OWASP detail pagina’s van risico’s, 2017)

In tabel 3.3 kun je aan de blauw gekleurde regels zien welke risico’s ik heb gekozen om te implementeren in de Webapplicatie scanner. Dit heb ik gedaan op basis van hoe vaak het risico voorkomt, en kijkend naar de moeilijkheidsgraad van de implementatie ervan, en ook op basis van de tijd die ik heb gekregen voor dit project. Mijn selectie bestaat uit A1 injecties, A3 XSS en A5 Security Misconfiguration. Deze zijn Vaak tot Heel Vaak voorkomend en zijn makkelijk tot gemiddeld implementeerbaar in het systeem. Wat ik ook heb opgemerkt is dat wanneer een Risico oftewel kwetsbaarheid makkelijk exploiteerbaar is, is de implementatie van het identificeren ervan ook makkelijk. Hier is een simpele reden voor, een webapplicatie scanner doet een poging om de kwetsbaarheid te exploiteren door een aanval te simuleren. Dit wordt in de meeste gevallen in een veilige test omgeving gedaan.

### 3.2.2 OWASP Selectie

### A1 injectie

Software ontwikkelaars gebruiken SQL queries om een actie uit te voeren op de database, deze queries hebben in sommige gevallen input nodig van de eindgebruiker, zoals bij een inlog formulier. De gebruiker voorziet de SQL Query met de benodigde argumenten, zoals de gebruikersnaam en wachtwoord. De applicatie bouwt met deze argumenten de query op en laat het uitvoeren door de database. In figuur 3.1 staat een voorbeeld van een veel gebruikte select query voor het authentiseren van gebruiker. De globale $\_POST[] variabelen zijn de argumenten die de eindgebruiker meegeeft als input aan de applicatie.



Figuur 3.1

SQL injectie (SQLi) is een applicatie security kwetsbaarheid dat ervoor zorgt dat cybercriminelen controle krijgen over de database van de applicatie. De cybercrimineel kan toegang krijgen tot of het veranderen en verwijderen van data. Dit gebeurt wanneer de applicatie onverwachte SQL commando’s opstuurt naar de server. De SQL injectie worden in de meeste gevallen via een web formulier of browserbalk ingevoerd als data. en als de server er niet in slaagt om het op de juiste wijze schoon te maken (sanitize)voor dat het de data toegevoegd wordt aan de SQL Query dan kan de aanvaller zijn eigen SQL commando’s eraan voegen wat de database zal uitvoeren.

Figuur 3.2

Met SQL taal kan de applicatie met de database communiceren. Hiermee wordt er data opgevraagd, gewijzigd of verwijderd.

SQL Injectie gebeurt wanneer de applicatie faalt om de ingevoerde data, via het formulier, schoon te maken van de SQL Query. Een aanvaller kan special opgebouwde SQL commando’s gebruiken die de database op verzoek van de applicatie laat uitvoeren.

Voorbeelden van een SQL Injectie in PHP

Voor het uitvoeren van een SQL Injectie moet de database en applicatie aan twee criteria voldoen, een relationele database dat SQL gebruikt, en een applicatie waar de eindgebruiker de controle heeft over de input die mee wordt gegeven aan een SQL-query. Voor meer informatie over SQL injectie refereer ik u naar Bijlage A.

### A3 XSS

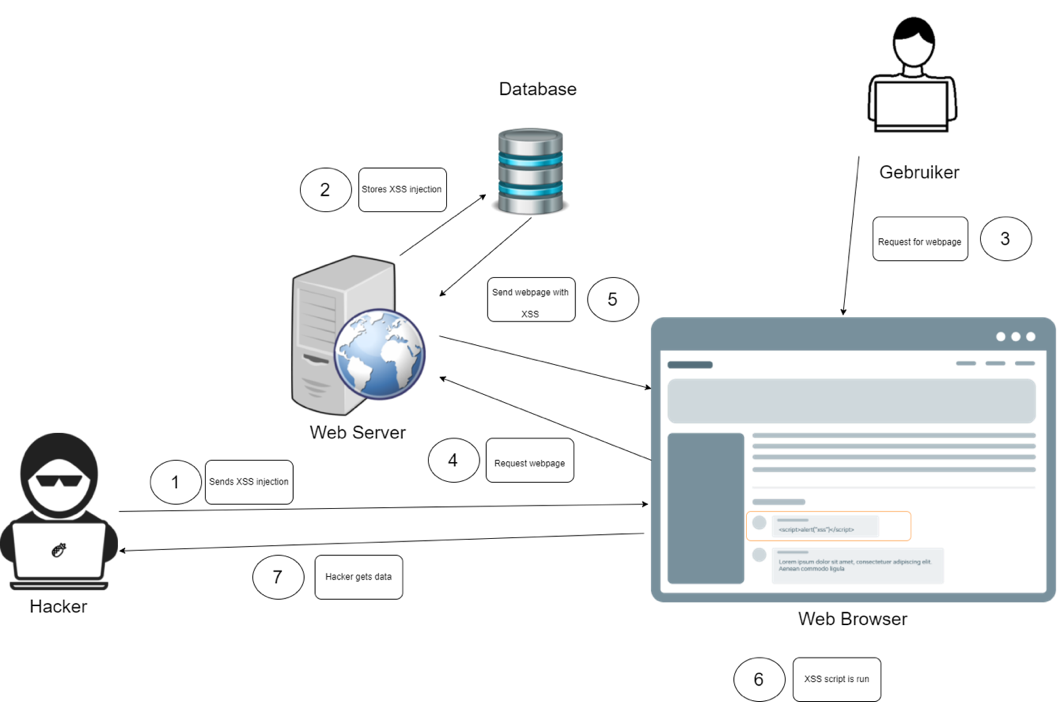
Cross-site scripting (XSS) is een client-side code injection aanval waarin de aanvaller/hacker kwaadaardige scripts uitvoert op webapplicaties. XSS wordt uitvoert in verschillende programmeertalen, VBScript, ActiveX, maar de meest misbruikte programmeertaal is Javascript. De primaire reden hiervoor is omdat de meeste webbrowsers afhankelijk zijn van Javascript. XSS is één van de meeste voorkomende web kwetsbaarheden en doet zich voor als een webapplicatie gebruikt maakt van ongecodeerde en niet gevalideerde gebruikersinput in de output dat het genereerd.



Figuur 3.3

Voor het verrichten van een XSS aanval moet de aanvaller/hacker een stukje javascript code plaatsen op de content pagina van een webapplicatie die de slachtoffer vervolgens bezoekt. De aanvaller/hacker moet ervoor zorgen de webbrowser van het slachtoffer de javascript code runt. De aanvaller/hacker kan dat op twee manieren doen, door social enginering technieken te gebruiken om het slachtoffer te overtuigen om de geïnfecteerde webpagina te bezoeken. Een ander manier is om een veel bezochte website, die gebruikers de functie geeft om zelf content te plaatsen zoals opmerkingen(comments), te infecteren met een XSS-injectie doormiddel van het plaatsen van een opmerking. De aanvaller/hacker hoeft alleen nog maar te wachten totdat de niets vermoedende gebruikers de website bezoeken. In figuur 3.4 wordt een voorbeeld gegeven hoe een HTML-script wordt uitgeprint op een webpagina.

**Cross-site scripting**



figuur 3.4

In figuur 3.4 wordt de uitvoering van een simpele maar effectieve XSS-aanval geïllustreerd. Er moet nog wel wat meer voor- en nawerk verricht worden om het rendabel te maken. De zeven stappen die ondernomen moeten worden zijn:

1. De aanvaller/hacker injecteert een kwaadaardige javascript code op de websites reactie pagina.
2. De server verwerkt de reactie en slaat het op in de database.
3. De gebruiker bezoek dezelfde reactie pagina via zijn webbrowser en maakt zichzelf hierdoor slachtoffer van de hacker.
4. De webbrowser doet een aanvraag voor de webpagina op verzoek van de gebruiker.
5. De server geeft de webpagina met alle reactie plus de reactie van de hacker terug aan de webbrowser.
6. De webbrowser runt de javascript code die het op de webpagina heeft gevonden.
7. De webbrowser stuurt de data van de niets vermoedende gebruiker naar de hacker.

Voor meer voorbeelden van XSS refereer ik u naar bijlage A.

### A5 Security Misconfiguration

### 3.2.3 WASC

De Web Application Security Consortium heeft als hoofddoel om standaarden te produceren voor web security. Zij en hun community zetten projecten op om dit doel te bereiken. Zoals eerder vermeld zijn er twee projecten relevant voor mijn onderzoek:

|  |
| --- |
| Project |
| WASC Threat Classification |
| Web Application Security Scanner Evaluation Criteria |

Het doel van het project WASC Threat Classification (bedreiging classificering) is om threats (bedreigingen) voor webapplicaties te classificeren. WASC deelt de dreigingen op in twee klassen/types: Aanvallen en weaknesses (kwetsbaarheden). Een lijst van de Threat Classes (bedreiging klasses) is te vinden op de website van WASC (wasc, bronnenlijst). Elke dreiging heeft een WASC ID waarmee deze geïdentificeerd kan worden en een detail pagina waar er een definitie wordt gegeven van de threat en voorbeelden. WASC gebruikt andere termen om threats beschrijven dan OWASP, bij OWASP zijn alle threats vulnerabilities (kwetsbaarheden).

Deze lijst zoals de “OWASP top 10 lijst” zal ik gebruiken in mijn software applicatie om kwetsbaarheden en aanvallen te kunnen identificeren en definiëren en zal dus in de datastructuur van mijn database model opgenomen worden. Niet alle aanvallen of kwetsbaarheden zijn die opgenomen zijn hebben een directe betrekking op de webapplicatie. Een aanval zoals Denial of Service (WASC-10) is een objectieve aanval op de server en niet de webapplicatie. Je zal een aantal threats tegenkomen die al in de OWASP deel zijn behandeld. Voor het Proof of Concept zal ik een aantal aanvallen van de WACS lijst met definitie opnemen in de webapplicatie. Deze zal ik in de tabel markeren.

De geïdentificeerde en gedefinieerde aanvallen en kwetsbaarheden zijn:

|  |  |  |
| --- | --- | --- |
| WACS ID | Threats type | Threat |
| WASC-42 | Aanvallen | Abuse of functionality |
| WASC-11 | Aanvallen | Brute Force |
| WASC-19 | Aanvallen | SQL Injection |
| WASC-12 | Aanvallen | Content Spoofing |
| WASC-18 | Aanvallen | Credential/Session Predicition |
| WASC-8 | Aanvallen | Cross-Site Scripting |
| WASC-9 | Aanvallen | Cross-Site Request Forgery |
| WAC-10 | Aanvallen | Denial of Service |
| WASC-15 | Weaknesses | Application Misconfiguration |
| WASC-16 | Weaknesses | Insecure Indexing |
| WASC-13 | Weaknesses | Information Leakage |
| WASC-01 | Weaknesses | Insufficient Authentication |
| WACS-02 | Weaknesses | Insufficient Athorization |
| WASC-14 | Weaknesses | Server Misconfiguration |

Tabel 3.4

WASSEC

Het doel dat WASC wilt bereiken met de Web Application Security Scanner Evaluation Criteria is een set van guidelines te definiëren voor het evalueren van webapplicatie scanners. Hiermee willen zij, de organisatie, security professionals begeleiden bij het evalueren van webapplicaties. Met de WASSEC worden een aantal kenmerken van een webapplicatie scanner geëvalueerd zoals het identificeren van kwetsbaarheden en de effectiviteit van de webapplicatie test. De WASSEC behandeld een variatie van componenten die deel uitmaken van een webapplicatie scanner zoals: de crawler/spider, parser (data verwerker), sessie manager, testing vaardigheid en de component dat het rapport genereerd.

Nu zal ik ook na het ontwikkelen van mijn Proof of Concept dit document gebruiken om op de correcte wijze een webapplicatie te evalueren, maar waar ik dit document primaire voor ga gebruiken is een requirement document waarin ik beschrijf aan welke eisen de Proof of Concept van de webapplicatie scanner moet voldoen.

Het document behandeld alle onderwerpen die nodig zijn bij het uitvoeren en evalueren van een webapplicatie scan. Het laat zien waar een webapplicatie scanner aan moet voldoen en het op een effectieve manier gebruikt kan worden. De categorieën die behandeld worden in dit document zijn:

|  |  |  |
| --- | --- | --- |
| Type | Categorie | Beschrijving |
| Proces | Protocol | Voor het testen van webapplicaties zal de scanner de communicatie protocollen moeten ondersteunen die gebruikelijk zijn. |
| Proces | Authenticatie | De ondersteuning van authenticatie voor het testen van webapplicaties. |
| Proces | Sessie management | Tijdens een scan is het belangrijk dat er een sessie onderhoudt wordt. |
| Component/tool | Spider/Crawler | Crawling is een actie die uitgevoerd wordt bij het parsen van een webpagina. Hyperlinks worden verwerkt door de crawler. |
| Proces | Parsing(HTML verwerking) | Vooraf van het scanproces is het nodig om een de website te verwerken, dit proces wordt door de webcrawler/spider gedaan. |
| Component/tool | Scanner/Tester | Het testen van website op kwetsbaarheden is de kernfunctie van een webapplicatie scanner. |
| Component/tool | Commando’s en controle | De commando’s waarmee de webapplicatie scanner wordt aangestuurd. Interactie tussen eindgebruiker en applicatie. |
| Component/tool | Rapporteren | Het genereren van rapporten op basis van de gevonden kwetsbaarheden. De resultaten kunnen dan buiten de interface van de applicatie bekeken worden. |

Tabel 3.5

In tabel 3.5 wordt een praktisch overzicht gegeven van waar een webapplicatie scanner uit bestaat en aan moet voldoen. Sommige van deze componenten en processen zitten in elkaar verweven zoals de Protocol, Authenticatie, Sessie management, parsing (HTML verwerking) deel uitmaken van de Spider/Crawler. De kern componenten van een webapplicatie scanner bestaan dus uit een GUI of CLI-interface, Crawler/Spider, Tester/Scanner, Rapport generator. Al deze componenten en processen zullen deel uitmaken van de Proof of Concept die ik in dit project zal ontwikkelen.

**Conclusie**

In het eerste deel van dit deelonderzoek heb ik gekeken naar welke kwetsbaarheden zich voor kunnen doen op een webapplicatie. Hierbij heb ik twee documenten bestudeerd, de OWASP top 10 lijst en de WASC Threat Classification. Na het bestuderen en analyseren van de informatie ben ik tot de conclusie gekomen om mij te richten op vier kwetsbaarheden die veelvoorkomende zijn en relatief makkelijk te detecteren zijn. Mijn selectie van kwetsbaarheden bevat de (A1/WACS-19) SQL injectie, (A3/WASC-8) Cross-site scripting en Security Misconfiguration (A5/WACS-13, 14 en 15). Voor het bewijzen dat mijn Proof of Concept werkt behoren deze vier kwetsbaarheden minimaal bij het testproces (Tabel 3.5, Testing) van de webapplicatie scanner. Het zijn kwetsbaarheden die vaak door hackers geëxploiteerd worden en hoge schade kunnen richten aan een bedrijf die afhankelijk is van een webapplicatie. Ook zijn deze twee kwetsbaarheden relatief makkelijk te implementeren. In het tweede deel van het deelonderzoek heb ik het WASC Web Application Security Scanner Evaluation Criteria document praktisch gebruikt om een algemeen web application scanner te ontleden zodat ik een overzicht kon maken van de componenten en processen waaruit het systeem in kwestie bestaat. Ik kan concluderen dat ik deze lijst (Tabel 3.5) zal gebruiken als referentie tijdens het maken van mijn requirement lijst oftewel MoSCoW.

## 3.4 Webapplicatie scanner

### 3.4.1 Inleiding

In de afgelopen decennia zijn er heel wat webapplicatie scanners op de markt verschenen, zowel commercieel als open source (gratis). Er zijn verschillende soorten webapplicatie scanners die ingezet kunnen worden op een variatie van vlaktes binnen een webapplicatie. Zo kunnen Webapplicatie scanners onderverdeeld worden in twee hoofdcategorieën Blackbox testing en Whitebox testing. Blackbox testing wordt geassocieerd met het testen van de buitenkant van een webapplicatie zonder kennis te hebben over de interne werking van de applicatie. Bij Whitebox testing wordt gerefereerd naar het testen vanuit de binnenkant van de applicatie. Dit heeft betrekking op het testen op kwetsbaarheden van de broncode. Vanuit deze twee categorieën hebben organisaties zoals OWASP tools ontwikkeld voor het detecteren van kwetsbaarheden vanuit de buitenkant en binnenkant van een web applicatie.

Webapplicatie scanners kun je volgens OWASP in vijf categorieën verdelen:

(https://www.owasp.org/index.php/Category:Tools\_Categories)

* Threat Modeling Tools
* Source Code Analysis Tools (SAST)
* **Vulnerability detection Scanning Tools (DAST)**
* Interactive Application Security Testing Tools (IAST)
* Penetration Testing Tools

Van de vijf categorieën zal ik er alleen maar één behandelen in dit document en dat is de DAST tools. De reden hiervoor is omdat ik voor het project een MVP zal ontwikkelen dat van de buitenkant, dus Blackbox testing, kwetsbaarheden detecteert van een web applicatie.

In dit deel zal ik onderzoek doen naar een aantal DAST tools om inzicht te krijgen in de werkwijze van de verschillende scanners. Ieder scanner verschilt in omvang, scan proces, software architectuur, het analytische vermogen, de wijze van rapporteren, etc.. Het is interessant om een aantal Webapplicatie scanners te bestuderen om zo hun werkwijze te leren kennen. Dit zal mij nieuwe inzichten bieden en zal mij helpen mijn eigen Webapplicatie scanner te verbeteren. Ook zal ik naar de fundamentele aspecten waaruit een Webapplicatie scanner gebouwd is.

OWASP heeft een lijst van 40 DAST gepubliceerd; deze lijst is op 26 juli ’17 voor het laatst geüpdatet. Deze lijst bevat zowel commerciële als open source tools. Ik zal er twee uit selecteren voor dit onderzoek, één zal commercieel zijn en de ander open source. Ik heb drie eisen opgesteld voor het selectieproces, 1) is dat beide voorloper moeten zijn op gebied van DAST tools en 2) het moeten goed gedocumenteerde tools zijn en als laatste 3) de tool moet makkelijk te verkrijgen zijn.

De eisen dienen als regels die het zoeken naar een geschikte Webapplicatie scanner vereenvoudigd.

Regels/eisen uitgelegd

1. Webapplicatie scanners die goed onderhouden worden met de laatste technologische vooruitgangen kan mij meer inzicht bieden dan een verouderde tool.
2. Goed gedocumenteerde tools bieden mij meer informatie aan over de werking van de tool voor mijn onderzoek.
3. Met tools die gemakkelijk verkrijgbaar zijn bedoel ik dat er geen hele registratie proces en goedkeuring vooraf moet gaan voordat ik er echt gebruik van kan maken.

Ik zal de beschikbare tools met deze drie regels/eisen valideren.

Uit het onderzoek van deel 3.4.2 heb ik een selectie gemaakt van twee DAST tools. Deze tools zal ik in de komende delen onderzoeken. Hiervoor zal ik de tools uitproberen en de werking documenteren, de documentatie doornemen en onderzoeken wat voor datastructuur zij gebruiken wat zij aan data verzamelen.

De scanners die ik voor dit deel heb gekozen zijn:

* Acunetix
* ZAP Zed Attack Proxy

Ik zal in dit onderzoek twee documenten behandelen: Acunetix handleiding en de brochure, beide zijn te vinden op de officiële website van Acunetix. Voor OWASP ZAP de documentatie die beschikbaar is gesteld op de index pagina van de officiële website waaronder de start gids van OWASP ZAP. De twee documenten bieden inzicht in de werking van beide DAST-tools.

Acunetix vermeld dat een DAST tool heeft drie kern processen:

**Crawl & Scan**

Voordat er gescand kan worden moeten er eerst resources worden opgehaald. De crawler zorgt daarvoor door een delen van een HTML pagina te scrapen. Scrapen is het proces van het selectief uitknippen van html elementen zoals de anchor tag <a>, wat als attribuut een hyperlink. Wanneer er voldoende gecrawld is kan een scan gestart worden.

**Detect & Alert**

Bij het detecteren van kwetsbaarheden is precisie dat telt. Tijdens de scan worden probeert de DAST-tool kwetsbaarheden in een webapplicatie te detecteren. Om te kunnen meten hoe goed en accuraat een DAST-tool kwetsbaarheden detecteert kan de OWASP Benchmark gebruikt worden.

**Prioritize & Manage**

Na een scan worden alle gevonden kwetsbaarheden geprioriteerd in een geïndexeerde lijst en gerapporteerd. De resultaten van de scans worden opgeslagen en kunnen herzien worden voor analytische doeleindes.

### 3.4.4 Acunetix

Volgens een document van Acunetic heeft 70% van alle websites een kwetsbaarheid dat kan leiden tot diefstal van gevoelige data [Acunetix document]. Acunetix maakt een paar goede punten in het document zoals, dat hackers zich concentreren op het volgende: Componenten van webapplicaties als winkelwagens, formulieren, login pagina’s en dynamische content. Dit is voor een grootdeel waar de webapplicatie scanner die ik ontwikkel op zal focussen. Een ander punt die Acunetix maakt is dat webapplicaties 24/7 toegankelijk zijn en dat zij controle hebben over kostbare data omdat zij, als het niet via een API gaat, in de meeste gevallen directe contact hebben met de backend, hoe anders moeten zij aan de data komen. Dit allemaal wordt in de eerste paragraaf toegelicht wat naar mijn mening een zeer goede inleiding is voor het document. Verder in het document vermeld Acunetix dat netwerk security amper bescherming biedt tegen webapplicatie aanvallen, dit komt doordat de netwerkpoorten 80/443 altijd openstaan. De reden dat deze poorten altijd openstaan is omdat zij iedere bezoeker moeten toelaten op de website. Anders heeft het geen nut om een website te hosten.

**Acunetix Web**

Zoals je kunt lezen is Acunetix een bedrijf dat zich bezighoudt met de security van webapplicaties. Hiervoor hebben zij een DAST-tool ontwikkeld. Acunetix heeft een desktopversie ontwikkeld en een web versie. Ik zal in dit deel de web versie behandelen. De Acunetix Web Interface is een gebruikersvriendelijke webapplicatie die de gebruiker na het inloggen naar een dashboard brengt. Vanaf het dashboard kunnen gebruikers vier managementtaken uitvoeren.

* Het configureren en beheren van websites die als targets (Doelwitten) worden ingevoerd
* Het starten van een security scan
* Het bekijken van statistieken
* Genereren van een rapport wat gebaseerd is op de kwetsbaarheden die gevonden zijn

Verder zijn er nog andere kleinere taken zoals het beheren van de gebruikers profiel. Targets zijn websites die de gebruiker als doelwit heeft geregistreerd voor de voorbereiding van een scan. Na het instellen van een target kan de gebruiker de scan starten. Voor het starten van de scan kan de gebruiker nog een aantal opties instellen. De opties zijn verdeeld over vier tabs, General, Crawl, HTTP, Advanced. In de tab General kan de type scan ingesteld worden, de snelheid van de scan en de logingegevens.

In de Crawl tab staan alle opties voor het instellen van de crawler, een crawler zorgt ervoor dat de benodigde informatie wordt verzameld voor de scan. Zoals de hyperlinks en formulieren. De HTTP tab is voor het authentiseren van een gebruiker tijdens de scan. De scan kan een login of registratieformulier tegenkomen tijdens de scan en weet dan met deze optie wat er ingevuld moeten worden. De laatste tab Advanced is ervoor de geavanceerde instellingen van de scan. Opties als technologie, Custom Headers, Custom Cookies en Allowed Hosts kan de gebruiker instellen om de scan te finetunen.

Na de scan kan de gebruiker navigeren naar het Scan Stats & Info scherm, hierop kan er naar de statistieken gekeken worden voor het analyseren van het resultaat, ook kan de gebruiker een lijst opvragen met alle uitgevoerde aanvallen en wat het resultaat daarvan is, dit vereist wel technische kennis over IT-security. Verder kan er nog naar de sitemap gekeken worden waar een lijst met alle gescande bestanden staan. Voor elk bestand staat er welke kwetsbaarheden gevonden zijn. Als laatste kan de gebruiker een rapport genereren op basis van de gescande resultaten. Bij het genereren van een rapport kan de gebruiker een template kiezen die de thema een doelgroep van het rapport bepalen. Acunetix is met al de opgenoemde features één van de voorlopers op gebied van webapplicatie scanners.

### 3.4.5 Zap Zed Attack Proxy

Zed Attack Proxy is een flagship project van Open Web Application Security Project. Het is een open-source security scanner tool dat onderhouden wordt door OWASP. Het is gratis te downloaden van de website van OWASP en ook biedt OWASP project samenwerking voor software ontwikkelaars die een bijdrage willen maken door te werken aan nieuwe features voor de applicatie. Het project is open-source wat betekent dat OWASP de broncode publieke heeft vrijgegeven. Het project kan worden gedownload op Github hiervoor het een softwareontwikkelaar geen toestemming voor nodig. Met een simpele git clone wordt er een kopie van het hele project op jouw computer geplaats.

De security scanner tool is special voor het testen van webapplicaties ontwikkelt. Eerder in dit document heb ik kort samengevat wat een security scanner in het algemeen doet. OWASP heeft in het document [OWASP ZAP 2.6 Getting Started Guide] dit onderwerp zelf behandelt. In het hoofdstuk Security Testing Basics definiëren zij security testen als “Software security testen is het proces van beoordelen en testen van systemen om security risico’s en kwetsbaarheden te ontdekken”. Er worden hier twee termen gebruikt die tot de basis taken behoren van een security scan tool: beoordelen en testen. Zij definiëren het beoordelen als het analyseren en ontdekken van kwetsbaarheden zonder de poging tot het exploiteren van deze kwetsbaarheden. Het term testen wordt gedefinieerd als het ontdekken en poging tot exploiteren van kwetsbaarheden. Beide taken hebben als doel kwetsbaarheden te ontdekken, maar het verschil zit in wat zij doen met de kwetsbaarheden die ontdekt zijn. To exploit or not to exploit.

In het hoofdstuk [Security Testing Basics] verdelen zij de basis van security testen in vier categorieën:

**Kwetsbaarheid beoordeling**

Het systeem is gescand en geanalyseerd voor security problemen.

**Penetratie test**

Het systeem ondergaat gesimuleerde aanvallen en het systeem wordt geanalyseerd.

**Runtime test**

Het systeem wordt geanalyseerd en het ondergaat een security test van een eindgebruiker.

**Code review**

De code van het systeem wordt gereviewd en het wordt specifiek geanalyseerd op security kwetsbaarheden.

ZAP-proxy

Het doel van het document is natuurlijk om een introductie te maken voor OWASP ZAP. Ik heb eerder al een aantal dingen verteld over de DAST-tool, maar ik zal er wat verder op in gaan in dit deel. Fundamenteel is ZAP een interceptie proxy tool, het staat tussen de eindgebruiker en webbrowser. De DAST-tool onderschept berichten die gestuurd worden door de webbrowser naar de eindgebruiker in dit geval de tester. Na de onderschepping worden de berichten door ZAP geïnspecteerd en als nodig aangepast, dit staat ook wel bekent als de ‘man in the middle’ aanval. Mocht er al een proxy in gebruik zal, vele bedrijven hebben dit, dan kan Zap verbinding maken met de gebruikte proxy.



Figuur 3.3.2.1

**ZAP client**

ZAP biedt voor alle grote besturingssysteem platformen een versie en is gemaakt voor zowel experts op gebied van cyber security en beginners. Zap biedt ook vele add-ons voor de ZAP DAST tool, deze zijn te vinden op de ZAP-marktplaats. ZAP wordt onderhouden door een grote gemeenschap die onderhoud verrichten en geregeld nieuwe add-ons (toevoegingen van features) ontwikkelen en publiceren op de marktplaats.

De ZAP client applicatie maakt gebruik van een User Interface waar de eindgebruiker de verschillende taken kan uitvoeren. Het design van de UI kwam mij bekent voor omdat het gebruik maakt van de Java Swing thema Nimbus. Dit betekent ook dat ZAP-proxy ontwikkeld is in Java. De ZAP UI bestaat uit 6 onderdelen: menubalk, takenbalk, boomstructuur venster, werkruimte venster, informatie venster, footer. Voordat er een pentest(penetratie test) uitgevoerd kan worden zal de proxy als eerst geconfigureerd moeten worden. De UI van Zap maakt het configureren zeer gemakkelijk, zoals het invoeren van een nieuwe SSL-certificaat of het instellen van een nieuwe proxy port.

De Zap client biedt verschillende scan configuraties voor het testen van een webapplicatie. De meest opvallende is de Snelle start test, deze test optie krijgt de eindgebruiker te zien wanneer de applicatie is opgestart. Om hiervan gebruik te maken voert de eindgebruiker de url in de tekstbalk en druk vervolgens op ‘Aanval’ om de test te starten. Er is wel een disclaimer, om een website te testen heb je wel toestemming nodig. De ZAP client zal, na het starten van de test, de website doorzoeken naar webpagina’s om deze vervolgens elke gevonden webpagina passief te scannen.

Na het passief scannen zal de ZAP client overgaan naar het actief scannen van de webpagina’s. Het doel van een passieve scan is om het voorwerk te doen voor de actieve scan. De passieve scan leest en neemt alle verkeer op dat tussen de browser en website wordt gecommuniceerd. Dit betreft de GET/POST requests en de responses ervan. Dit is de wijze waarop een webbrowser client communiceert met de webserver, door request (een verzoek voor een webpagina) en een response (antwoord op het verzoek). Na dit proces analyseert ZAP client de data en kijkt of er ‘known issues’ (bekende problemen) gevonden zijn. Actief scannen is meer gericht op het aanvallen van de gevonden ‘known issues’. Bij het actief scannen worden er echte aanvallen uitgevoerd, dat betekent dus dat het doelwit risico’s kan lopen. Dit is ook een reden waarom je eerst toestemming moet hebben voordat je een test mag uitvoeren. Voor meer informatie over de ‘known issues’, deze worden in het hoofdstuk OWASP top tien lijst behandeld.

**Test omgeving**

Voor het uitvoeren van een scan heb ik een test omgeving opgezet. De test omgeving bestaat uit drie componenten: de browser, de proxyserver en de webserver waar DVWA op draait. De hele test omgeving zit in een lokaal netwerk.

Figuur 3.3.1 - testomgeving

### 3.4.7 Conclusie

****Uit de testen is gebleken wat de werking is, wat voor datastructuur beide DAST tools gebruiken en wat voor data zij verzamelen. Fundamenteel zijn er veel gelijkenissen, de richtlijnen van de standaarden volgen van WASC (H 3.2.3, WASC). Waar de beide DAST tools in verschillen is de hoeveelheid informatie zij vrijgeven tijdens het scannen en de hoeveelheid data dat zij opslaan. Wat mij opviel was dat Acunetix veel schoner en eleganter is dan OWASP, Acunetix laat tijdens het scannen in het dashboard overzichtelijke en informatieve statistieken over de vooruitgang van de scan.

Figure 3.4.1

OWASP is meer verbose tijdens het scannen en laat een tabel zien waar in een vlug tempo, alle URL’s waar aanvallen op verricht worden zien, dit zonder enige context te geven over de vooruitgang van de scan, behalve dan de procesbalk. Een voordeel dat alle getoonde informatie biedt, is dat het als data kan dienen in een Excel sheet waaruit analyses gemaakt kunnen worden.



Figure 3.4.2

Wat betreft de datastructuur die beide tools hanteren en de data dat verzameld wordt zijn ook verschillen te vinden. Kenmerkend voor OWASP is de hoeveelheid data dat verzameld wordt. De data wordt op een onoverzichtelijk manier gepresenteerd in de applicatie. Dit maakt deze tool minder gebruiksvriendelijker dan Acunetix. Waar Acunetix 4 tabellen gebruikers genoeg data om er nuttige informatie van te maken, gebruikt OWASP 6 tabellen met meer als genoeg data. Omdat ik een MVP maak is het van belang dat ik alleen essentiële data verzamel en het niet te druk maak.

**Datastructuur voor een MVP**

Een MVP heeft informatie nodig over de volgende vijf onderwerpen, als je er één zou weghalen dan kun je geen Webapplicatie scanner bouwen.

De tabellen

|  |
| --- |
| Website |
| URL |
| Server |

|  |
| --- |
| Headers |
| Key (Sleutel) |
| Value (waarde) |

|  |
| --- |
| Link |
| Methode |
| URL |

|  |
| --- |
| Scan |
| Module naam |
| Risico |
| Parameter |
| Aanval |

|  |
| --- |
| Rapport |
| Naam |
| Bestand |

## 3.5 Technologie

Deelvragen

Wat zijn de tools en technieken die ik kan gebruiken bij het ontwikkelen van een Webapplicatie scanner?

### 3.5.1 Web Application Framework

Een webapplicatie ontwikkelen met enkel een programmeertaal (e.g. PHP, Java, Python) kan een lastige klus zijn waarin veel tijd en als het voor een klant is, geld ingestoken wordt. Frameworks zijn net jetpacks voor een programmeertaal (Upwork, 2017), zij versnellen en vermakkelijken het ontwikkelproces. Dat doen de web frameworks met behulp van webservices (e.g. depandacy managers), web resources (e.g. config files), en web APIs (e.g. ORM) die zij out-of-the-box aanbieden. Het is net als het bouwen van een schip, alle bouwcomponenten worden voorgemaakt aangeleverd zodat het bouwproces versnelt kan worden.

De Core features van Webapplicatie Frameworks zijn:

Libraries – Herbruikbare, voorgeprogrammeerde code dat softwareontwikkelaar gebruiken als bouwblokken voor het ontwikkelen van softwareapplicaties.

API – Application Programmer Interface, zijn kleine programma’s die de software ondersteunen in het uitvoeren van functionaliteiten.

Cashing – Het opslaan van web resources in het tijdelijke geheugen dit verlaagt de server workload en gebruik van brandbreedte.

URL Mapping – Een systeem waarmee je URL’s kunt koppelen aan zelf gemaakte URL’s

Security – Sub Frameworks voor authenticatie en autorisatie

**PHP Framework Benchmark**

Techempower brengt eerder jaar een benchmark uit voor Web Frameworks. Zij testen de Web Frameworks op een aantal types JSON serialisatie, enkele query, data updates. Ik zal de PHP Frameworks met elkaar vergelijken op performance. Uit het resultaat zal ik nog niet concluderen voor welke Web Framework ik ga kiezen. Hiervoor zal ik ook kijken naar de features en schaalbaarheid die de Frameworks bieden. PHP Frameworks zijn over het algemeen traag en zullen niet veel verschillen in snelheid.

**PHP Frameworks**

* Phalcon
* Slim
* YII2
* Laravel
* Symfony 2
* Zend

In tabel 3.7.1 zal ik de benchmarks van techempower.com tonen om de keuze die ik heb gemaakt te onderbouwen. De PHP Frameworks worden op 3 test types getest:

* JSON serialisatie - In deze test is elke response een JSON serialisatie van een geïnstantieerd object dat de key value koppelt aan de waarde Hello, World! {“message”:”Hello, World!”}
* Enkele query – In deze test wordt elke request verwerkt door het ophalen van een enkele regel uit een simpele database. {“id”:3217,”randomNumber”:2149}
* Data updates -

|  |  |  |
| --- | --- | --- |
| PHP Framework Benchmark (Gesorteerd op Performance) | | |
| JSON serialisatie | | |
| Framework | Performance (hoe hoger hoe beter) | Wachtijd (hoe lager hoe beter) |
| Phalcon | 39,865 (7.1%) | 50.0 ms (3.9%) |
| Yii2 | 10,388 (1.9%) | 27.1 ms (2.1%) |
| Zend | 9,784 (1.7%) | 31.1 ms (2.4%) |
| Slim | 9.640 (1.7%) | 34.3 ms (2.7%) |
| Laravel | 7,020 (1.3%) | 38.0 ms (3.0%) |
| Symfony 2 | 3,588 (0.6%) | 74.3 ms (5.9%) |
| Enkele query | | |
| Phalcon | 22,604 (10.8%) | 29.4 ms (1.0%) |
| Slim | 9,779 (4.7%) | 29.3 ms (1.0%) |
| Yii2 | 6,789 (3.2%) | 39.6 ms (1.4%) |
| Laravel | 5,385 (2.6%) | 51.9 ms (1.8%) |
| Zend | 3,965 (1.9%) | 67.9 ms (2.4%) |
| Symfony 2 | 2,508 (1.2%) | 105.4 ms (3.7%) |
| Data updates | | |
| Slim | 768 (17.5%) | 330.0 ms (5.7%) |
| Zend | 766 (17.5%) | 330.4 ms (5.7%) |
| Yii2 | 762 (17.4%) | 332.2 ms (5.8%) |
| Phalcon | 642 (14.7%) | 398.4 ms (6.9%) |
| Symfony 2 | 321 (7.3) | 490.8 ms (8.5%) |
| Laravel | - | - |

Tabel 3.5.1 – benchmark

**Performance reviews**

**Phalcon**

Phalcon scoort verre weg het hoogst met het uitvoeren van JSON serialisaties en Enkele queries. Phalcon is een PHP-extensie geschreven in C, in 2012 uitgebracht en is gebouwd voor snelheid zoals je kunt terugvinden in de benchmark onderzoek.

**Slim**

Slim eindigt als tweede in het benchmark van techempowerd maar, scoort het hoogst op Data Updates wat een best zwaar belanden opdracht is. Slim is een Micro Framework met een lage leerdrempel heeft**.** Slim is het best geschikt voor kleine, simpele projecten of voor het bouwen van een simpele, flexibele API.

**YII2**

YII2 is een Framework dat OOP en DRY gebruikt als programmeer concepten. Het heeft veel features out-of-the-box voor het snel ontwikkelen van een moderne website. Het is geschikt voor het bouwen van complexe webapplicatie als CMS’s en CRM’s.

**Zend**

Zend is een robuste en stabiele PHP Framework met dat al 9 jaar oud is, vele grote bedrijven gebruiken Zend voor hun webprojecten. Zend is een PHP Framework die niet geschikt is voor klein tot middelgrote webprojecten.

**Symphony2**

Wat voor Zend geldt, geldt ook voor Symphony. Het is een PHP Framework dat bedoeld is voor grote complexe enterprice projecten. Het heeft een grote set van herbruikbare componenten. Veel van deze componenten worden door andere Web Frameworks gebruikt zoals Laravel. Door de omvang is het jammer genoeg ook een trage framework zoals te zien is in tabel 3.7.1.

**Laravel**

Laravel is een relatief new PHP Framework, geïntroduceerd in 2011 en volgens een Sitepoint enquête (Sitepoint, 2015), verreweg het meest populaire, Symphony is tweede geëindigd. Laravel biedt een essentiële set van features voor zowel kleine als grote projecten. Het heeft een actieve onlinegemeenschap die ondersteuning bieden op gebied van Laravel applicatieontwikkeling. De out-of-the-box features van Laravel zijn o.a. Authenticatie, Artisan Console, RESTfull Routing, Migration, ORM, Composer depandancy manager, CLI-support, Blade. Laravel is misschien niet de snelste PHP Framework, dit komt grotendeels omdat het componenten uit Symphony gebruikt, maar door de uitgebreide toolbox, grote supportgemeenschap en makkelijk te leren Framework, maakt Laravel geschikt voor iedere PHP-project.

**Conclusie**

**Laravel**

Voor dit project (Webapplicatie scanner) is iedere PHP Framework geschikt, er zullen geen problemen zijn met de performance simpel weg omdat ik een Minimal Vaible Product ga maken wat klein in schaal is. Wat wel belangrijk is voor het project is dat de PHP framework de specifieke tools die ik nodig heb voor het ontwikkelen van de Webapplicatie scanner bevat. Laravel is de PHP Framework die mij de tools, flexibiliteit en ondersteuning kan bieden om van dit project een geslaagd project te maken. Omdat Laravel RESTfull Routing als feature heeft is een andere Framework zoals Slim hoeft dus niet gebruikt te worden.

### 3.5.2 Database

Het kiezen van een web framework is een belangrijke stap richting het ontwikkelen van een web applicatie een bijna zo even belangrijke stap is het kiezen van een databasemanagementsysteem of in het kort DBMS. Een DBMS is in de meeste gevallen open source dus gratis voor gebruik, zo ook MySQL en PostgrSQL. Een database zorgt voor de opslagmogelijkheden van data. Data dat geproduceerd wordt door een web applicatie, iets wat een Webapplicatie scanner wel degelijk doet (H, 3.3). Het verzamelen en opslaan van website hyperlinks kan een accumulatief proces zijn. Bij een crawl proces kunnen er wel duizenden hyperlinks verzameld worden, in een rap tempo. Maar niet alleen dat tijdens een scan kunnen er duizenden aanvallen uitgevoerd worden, de data hiervan zal ook moeten worden opgeslagen. Kortom een DBMS is een zeer cruciaal component van het gehele systeem.

Er zijn over het algemeen twee type databasemanagementsystemen SQL (Structured Query Language) en NOSQL (Not Only SQL). SQL wordt al meer dan 4 decennia in de professionele werkveld gebruikt en het gebruik is in de jaren 90 exponentieel gestegen met de komst van web applicaties en open source optie als MySQL en SQLite. NOSQL daarentegen bestaat al sinds de jaren 60 maar wordt pas recentelijk gezien als een waardige DBMS met de komst van MongoDB, CouchDB, Redis en Cassandra. De vraag is dus welke van de twee DBMS soorten is het meest geschikt voor de Webapplicatie scanner?

**SQL of NOSQL**

NOSQL is geen nieuwe technologie dat SQL vervangt, het gebruikt een ander manier van dataopslag. Geen is beter, het hangt af van de type applicatie die gebruikt maakt van een DBMS. Voor sommige applicaties is SQL meer geschikt en voor andere NOSQL.

In figuur 3.5.2 zijn de twee soorten databases geïllustreerd, links een relationele database en rechts een non-relationele database. De relationele database heeft een tabel structuur met kolommen en rijen. De verbinding lijn tussen de beide tabellen noem je een relatie. In deze instantie is het een één op veel relatie. De hele tekening noem je een Schema wat de blauwdruk is van een database.

De non-relationele database slaat data op in een document bestand. De data heeft een key-value structuur en wordt in de meeste gevallen geschreven in JSON annotatie. NOSQL is bedoelt voor een groot hoeveelheid, ongestructureerde data. Met dit soort data is het moeilijk om een duidelijk gedefinieerde schema te maken. NOSQL maakt geen gebruik van relaties maar van Collecties van data. Als er Comments voor een specifieke blog opgehaald moet worden dan moet je dat specificeren in de Query door gebruik te maken van indexen, of sub arrays.

Figuur 3.5.2.1



**CONCEPT**

**Deelvraag**

**Hoe ziet het concept eruit van de webapplicatie scanner?**

# 4 Concept ontwikkeling

**Disclosure**

Voor ik begin aan het concept documentatie wil ik melden dat ik geen Dynamic Application Security Testing (DAST) tool ga ontwikkelen voor de proof of concept. Ik heb in mijn vooronderzoek twee DAST-tools onderzocht, dit heb ik gedaan omdat 1) DAST-tools voorlopers zijn op het gebiedt van webapplicatie scanners 2) omdat zij de twee DAST-tools alle fundamentele kenmerken hebben van een webapplicatie scanner. Ik zal mij voor het concept focussen op deze fundamentele kenmerken.

**Inleiding**

Dit is het volgende stap in het ontwikkelproces van een softwareapplicatie, het ontwikkelen van een concept. Hiervoor heb ik een onderzoek gedaan naar de theorie van een webapplicatie scanner met doel om erachter te komen wat de benodigdheden zijn. In dit deel zal ik de concepten van alle componenten die deel uitmaken van het systeem beschrijven. Van dit punt zal ik naast het woord softwaresysteem ook de naam SecurityRapport gebruiken om naar het softwaresysteem te refereren.

Om een terugkoppeling te maken naar hoofdstuk 1 (Inleiding van de scriptie) wil ik het primaire hebben het doel van de webapplicatie scanner. Het doel van dit project is om een Proof of Concept te ontwikkelen van een webapplicatie scanner. Dit in de vorm van een MVP (Minimal Viable Product), wat een versie van een softwareapplicatie is, die de aan de minimale eisen voldoet om te kunnen functioneren als webapplicatie scanner. Deze soort producten zijn vooral aanbevolen voor als een onderneming wilt experimenteren met een nieuwe ideeën. Omdat het een grote investering zou, zijn mocht het echt ontwikkeld worden, is het een praktische keuze geweest om eerst te gaan experimenteren met het idee van een geïntegreerde webapplicatie scanner. Zo is ook uit mijn onderzoek gebleken dat het implementeren van een gelijksoortig systeem kostbaar kan zijn mocht het door een derde partij gedaan worden. Mijn eigen quote uit hoofdstuk 1 geeft aan dat het aanbieden van een gratis webapplicatie scanner zorgt voor een elegante oplossing voor dit financiële probleem.

“Een gratis webapplicatie scanner dat makkelijk te gebruiken is voor een bestaande klant van S5 zou ideaal zijn om de kostenpost van consultancy en dure licentie te omzeilen”.

**Concepten die behandeld worden**

Het systeem bestaat uit kern componenten en ondersteunende componenten. Ik zal mij in dit deel primaire focussen op de kern componenten, zo kan ik de rol van de ondersteunede componenten beter omschrijven. Alle componenten en processen die in hoofdstuk 3 (Onderzoek, WASC) voorkomen zal ik in dit deel behandelen. Ook zal ik uitleggen welke rollen de verscheidenen technologieën hebben voor elke component of proces.

**Kern componenten**

De kern componenten zoals deze in WASC WASSEC document aangegeven zijn, geven een overzicht van de onderdelen waar een webapplicatie minimaal uit moet bestaan. Binnen het project context komt er nog één kern component bij, de database. Kern componenten zijn essentieel voor de werking van de applicatie, waarom de database een essentiële componenten is zal ik later in het document beargumenteren.

|  |  |
| --- | --- |
| Type | Categorie |
| Component/tool | Crawler |
| Component/tool | Scanner |
| Component/tool | Command Line Interface |
| Component/tool | Rapporteren |
| Component | Database |

Tabel 4.1

**Ondersteunende componenten**

Dit zijn componenten die het de kern applicatie bruikbaar maken binnen de project context waarin ik dit systeem ontwikkelen. Een deel van het project context is: de klanten van S5 de mogelijkheid geven om een scan aan te vragen doormiddel van een CMS-extensie. Zonder deze ondersteunende componenten kunnen de klanten geen gebruik kunnen maken van de webapplicatie scanner. De kern applicatie is niet afhankelijk van deze ondersteunende componenten maar het projectdoel wel.

|  |  |
| --- | --- |
| Type | Categorie |
| Ondersteunend comp. | API |
| Ondersteunend comp. | WordPress extensie |
| Ondersteuned comp. | Email service |
| Ondersteunend comp. | Admin panel |

Tabel 4.2

**Processen**

Naast de componenten zijn er ook processen die uit worden gevoerd op verschillende momenten van het system cyclus. Ieder component heeft processen waarvan een aantal beschreven staan in het WASC WASSEC document. Processen zijn weer onderverdeelt in functies die een beschrijving geven van de verschillende operaties die een proces uitvoert. Een aantal van deze functies zal ik in de realisatie deel omschrijven. Ondersteunende componenten hebben ook processen. Deze verschillen veel van elkaar omdat er gebruik wordt gemaakt van verschillende frameworks.

|  |  |  |
| --- | --- | --- |
| Type | Component | Categorie |
| Proces | Crawler/Spider | HTTP communicatie |
| Proces | Crawler/Spider | Sessie management |
| Proces | Crawler/Spider | Authenticatie |
| Proces | Crawler/Spider | Parsing (HTML verwerking) |
| Proces | Crawler/Spider | Data opslag |
| Proces | Scanner | Aanvallen (meerdere types) |
| Proces | Scanner | Data opslag |
| Proces | Commnado’s en Controle | Scan commando |
| Proces | Commnado’s en Controle | Scan configuratie |
| Proces | Rapporteren | Rapport genereren |
| Ondersteunend proces | Rapporteren | Verzenden van rapport |
| Ondersteunend proces | API | API calls |
| Ondersteunend proces | CMS extensie | Formulier verwerking |
| Ondersteunend proces | Admin panel | Activeren van accounts en het verzenden van rapporten |

**Tabel 4.3**

## 4.1 Concept van het system (SecurityReport)

Het systeem wordt een geautomatiseerde Webapplicatie scanner die er moet voor zorgen dat de klanten van S5 kosteloze scans kunnen laten uitvoeren op hen webshop. Het systeem bestaat uit vier subsystemen die in dit deel worden behandeld.

|  |
| --- |
| Subsystemen |
| CMS extensie |
| API |
| Webapplicatie scanner |
| Database |

**Tabel 4.4**

Deze vier subsystemen worden geïntegreerd in het operationele omgeving van het systeem (Operationele Diagram) waar ieder zijn functionele doel uitvoert.

**Systeem lagen**

In het diagram 4.1 kun je goed zien wat voor informatica model het gebruikt, hoe het systeem is opgebouwd, van hoeveel servers het gebruik maakt, uit hoeveel lagen het bestaat en hoe de subsystemen met elkaar communiceren.



Figure 4.1 vergrote diagram bijlage B

|  |  |
| --- | --- |
| **Systeem Specificatie** | |
| **Technology** | **PHP, SQL** |
| **Frameworks** | **Laravel** |
| **Sub systemen** | **4** |
| **Systeem lagen** | **6** |
| **Communicatielijnen** | **3** |

**Tabel 4.5**

**CMS extensie**

Het doel van de CMS extensies is om het systeem met een “endpoint” te voorzien om een brug te slaan tussen de klant en het systeem. Met de CMS extensie heeft de klant de mogelijkheid om een aanvraag te doen op een scan. Het CMS biedt hiervoor een formulier aan. Ook maakt het deel uit van de registratie proces van een klant.

**API**

De API is ervoor om het systeem te voorzien met de communicatie tussen de CMS extensie en Webapplicatie scanner. Essentieel zal het alleen maar zorgen dat de aanvraag via een HTTP REST call door gecommuniceerd kan worden naar de Webapplicatie scanner.

**Webapplicatie scanner**

De Webapplicatie scanner staat centraal in het systeem. Dit subsysteem zorgt ervoor dat de webapplicatie, die opgegeven is voor een scan, gescand wordt om kwetsbaarheden binnen de web applicatie te identificeren. Hierop volgt nog een voorproces dat het systeem met data voorziet zoals de hyperlinks waarmee het de gesimuleerde aanvallen mee uitvoert. In dit proces wordt het doelwit gecrawld voor data. Crawlen is het schrapen van HTML documenten met als doel specifieke HTML element te vinden.

Na het afronden van de crawl proces worden de gesimuleerde aanvallen op het doelwit uitgevoerd. Dit doet het doormiddel van Blackbox Testing (H 3.4.1) wat een techniek is die van buitenaf een software applicatie probeert open te breken, zonder kennis te hebben over de interne werking van het doelwit. Er wordt naar twee soorten kwetsbaarheden gezocht (H 3.2.1) SQL injectie en Cross-site scripting (XSS), deze twee kwetsbaarheden behoren tot de basis van bijna iedere scan, zijn veelvoorkomend en hebben een hoge risico factor (OWASP, H 3.2.1). Naast deze twee kwetsbaarheden wordt er ook gescand op Security Misconfiguraties. Deze kwetsbaarheid wordt kan onderverdeelt worden in die soorten: Information Leakage (WASC-13), Server Misconfiguration (WASC-14), Application Misconfiguration. Hiervoor worden er geen gesimuleerde aanvallen uitgevoerd, de webapplicatie scanner scant de gecrawlde Headers op deze kwetsbaarheden. Een andere taak die de Webapplicatie scanner uitvoert is het genereren van een rapport dat gebaseerd is op de geïdentificeerde kwetsbaarheden en het verzenden van het rapport naar de klant toe via de email.

Het subsysteem bestaat als enige uit meerdere lagen. De drie lagen zijn de service laag: service laag, logica laag en de data toegang laag. Ieder laag heeft een eigen rol binnen het sub systeem en zorgt ervoor dat de data goed wordt verwerkt.

Figure 4.2

|  |  |
| --- | --- |
| Laag | Omschrijving |
| Service laag | Deze laag zorgt voor dat de data opgeslagen, geüpdatet en opgehaald kan worden. (CRUD, Glossary, 16) |
| Logica laag (controller) | Deze laag maakt gebruik van de service laag om de data te verwerken. |
| Data toegang laag (Model, ORM) | Service laag maakt gebruik van deze laag om toegang te verkrijgen naar de database doormiddel van een ORM (Glossary, 17) |

Tabel 4.6

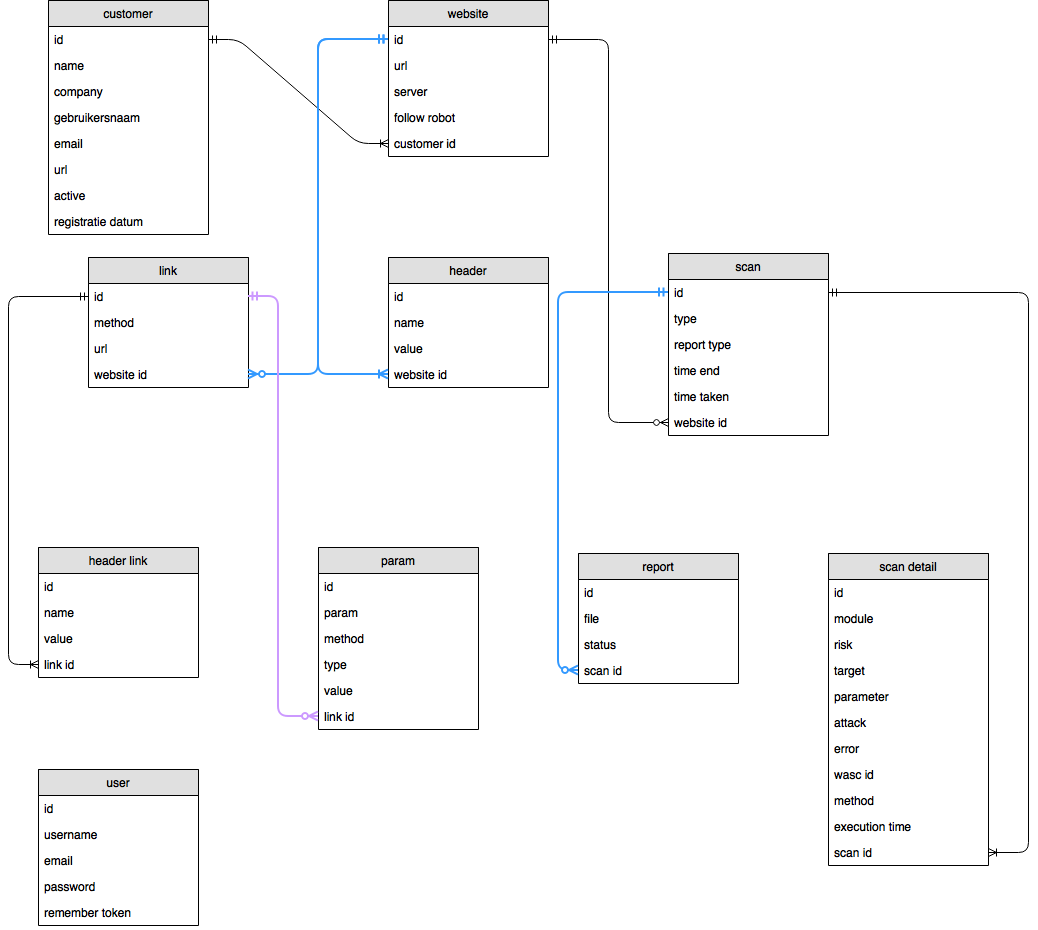
**Admin Panel**

De Admin panel maakt deel uit de webapplicatie scanner en maakt gebruik van de Service laag en de Logica laag. Met de Admin panel kunnen er twee belangrijke functies worden uitgevoerd die gezien kunnen worden als ondersteunende functies. De eerste functie behoort tot de registratie proces en wordt gebruikt om de klanten account te activeren. De tweede behoort tot de email service, tijdens het verzenden kan het zo zijn dat er iets mis is gegaan zoals het wegvallen van het internet. Om dit probleem op te lossen kan de administrator de email vooralsnog opsturen via de Admin panel.

De database is de vierde en laatste van de essentiële componenten. Selectieve data dat de Webapplicatie scanner genereerd wordt opgeslagen in de database. De datastructuur is gebaseerd op mijn vooronderzoek naar de datastructuren die OWASP ZAP en Acunetix hanteren (H 3.4.6). Uit dat onderzoek heb ik alleen de essentiële entiteiten en attributen eruit kunnen halen. Tijdens het ontwikkelen zijn er vele aanpassingen gedaan aan de datastructuur om de data die de webapplicatie scanner te accommoderen. In figuur 4.4 staat een versimpelde versie van de ERD.

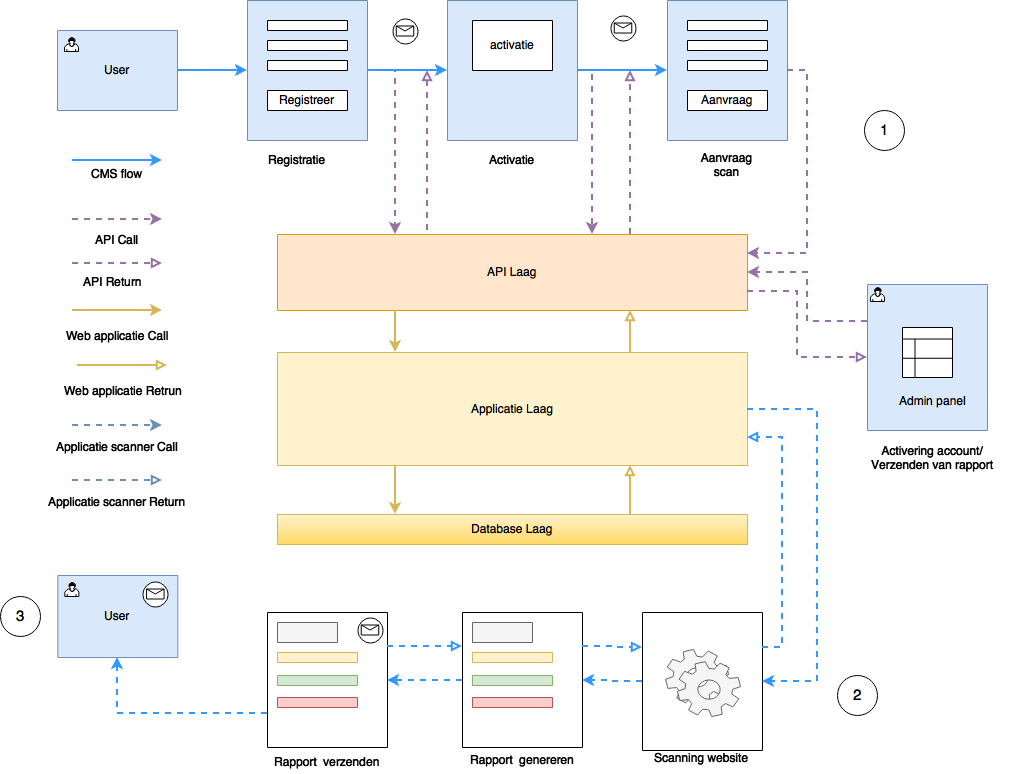
|  |  |  |
| --- | --- | --- |
| Entiteit | Relatie | Entiteit |
| Customer | Een op veel | Website |
| Website | Een op 0 of veel | Link |
| Website | Een op veel | Header |
| Website | Een op 0 of veel | Scan |
| Link | Een op veel | Header Link |
| Link | Een op 0 of veel | Param (parameter) |
| Scan | Een op Veel | Scan Detail |
| Scan | Een op 0 of veel | Report |
| User | n.v.t. | n.v.t. |

Tabel 4.7

****

Figuur 4.3

**Workflow van het systeem**



FIGUUR 4.4

De workflow van het softwaresysteem is op het CMS-extensie en het administratieve paneel geheel geautomatiseerd. Het gehele proces staat in figuur 4.5 geïllustreerd en bestaat uit drie stappen.

**Registratie, activatie en aanvraag**

In stap (1) begint het proces met de registratie van een klant. In dit scherm wordt er naar de naam en bedrijf van de klant gevraagd, dit zijn twee verplichte velden. Met deze twee velden wordt ook de accountgegevens van de WordPress website meegestuurd (dit wordt in de paragraaf CMS extensie verder toegelicht H 4.2). Al deze data wordt verstuurd naar de API laag die het weer door communiceert naar de Applicatie laag, deze laag verwerkt alles in de database. Nu het account is geregistreerd krijgt de gebruiker een email ter bevestiging en wordt de tweede scherm in het proces getoond. Hierin staat een bericht over de handelingen die verricht moeten worden door de administrator met betrekking tot het activeren van het account en de eis waar een gebruiker aan moet voldoen om gebruik te kunnen maken van SecurityReport. De eis is simpel, ben je een klant dan ben je gemachtigd om gebruik te maken van SecurityReport. Hierop volgt een controle van de administrator die de registratie zal moeten valideren. Dit kan worden gedaan door contact op te nemen met het bedrijf dat geregistreerd is en te kijken of het bedrijf in het klantenbestand voor komt. Als het blijkt dat het om een klant gaat dan activeert de administrator het account, na het activeren wordt er een bevestigingsemail naar de klant gestuurd. De klant is op dit punt gemachtigd om een scan aan te vragen, dit behoort tot de laatste fase van deze stap. Een scan wordt aangevraagd door het invullen van een formulier, dit formulier wordt in het hoofdstuk H 4.2 uitgelegd. Na het invullen van het formulier en het verzenden ervan zal stap 2 in gaan gezet worden, hierin wordt er o.a. de website gescand en wordt er een rapport gegenereerd.

**Scannen en rapporteren**

In stap (2) worden er twee processen uitgevoerd als eerst het scannen van de webapplicatie, wat in de context van dit systeem altijd een WordPress website is. In een standalone omgeving kan het systeem ook webapplicaties buiten de Wordpress omgeving scannen. Het scannen van de WordPress webapplicatie is zoals in de inleiding van H 4.1 is beschreven onderverdeelt in een aantal subprocessen. In H 4.4 wordt dit onderwerp uitgebreid behandeld. De subprocessen zijn: Crawling en Scanning wat behalve voor de Header Module (H 4.4.3), gesimuleerde aanvallen zijn. Het crawlen en scannen zorgen een groot deel van de data dat gebruikt wordt voor het genereren van het rapport (H 4.4.4). Het tweede proces is genereerd een rapport op basis van de gevonden kwetsbaarheden. Dit rapport wordt van een HTML document naar PDF formaat geconverteerd (H 4.4.4), dit is het meest gangbare manier om in de taal PHP een PDF op te bouwen in de markup taal HTML en op te maken in de styling taal CSS. Na het genereren van het rapport wordt het opgeslagen in een netwerk map die voor de administrator via de admin panel toegankelijk is.

**Het verzenden en beheren van rapporten**

In deze laatste stap (3) wordt het gegenereerde rapport verstuurd naar de klant via de mailserver van MailGun dit is tevens ook de afsluiting van het gehele proces. Mocht het rapport om bijvoorbeeld een netwerkprobleem niet verstuurd zijn dan kan de administrator de lijst met niet verstuurde rapporten ophalen en deze nogmaals opsturen wanneer de problemen zijn verholpen. Bij de ontvangst van het rapport door de klant is de service tot een eind gekomen en kan de klant opnieuw een scan aanvragen.

## 4.2 CMS extensies

** CMS**

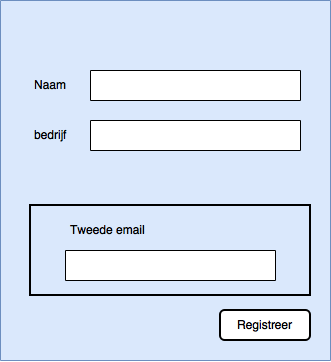
Content Management Systeem

Een CMS is het systeem achter een website dat het beheren hiervan eenvoudiger maakt. Zo kan iemand met relatief weinig technische kennis een website bouwen en beheren.

Business Doel: Een Client applicatie voor de klant wat dient als Endpoint

Het doel van het WordPress extensie is om de klant met een interface te voorzien waarmee een aanvraag voor een scan gedaan kan worden. De WordPress extensie wordt op de online extensie markt van WordPress beschikbaar gesteld zodat elke klant er makkelijk aan kan komen. De WordPress extensie bestaat uit drie schermen de registratie scherm, de activatie scherm en de aanvraag scherm. Deze schermen zorgen ervoor dat de klant de hele Customer Journey kan belopen.

**Registratie**



Figuur 4.5

Bij het starten van de extensie is het, het registratieformulier die als eerste wordt getoond aan de klant. Het registratieformulier bestaat uit drie velden naam, bedrijf en tweede email. De naam en bedrijf zijn verplichte velden omdat deze gebruikt worden voor het identificeren van een klant. De tweede email dient als alternatieve email, de extensie maakt gebruik van de email die geregistreerd staat in de WordPress account.

De WordPress extensie stuurt een deel van de accountgegevens mee deels omdat de gebruiker dat dan niet hoeft mee te geven en voor het authentiseren van een account, dit behandel ik in een ander paragraaf. Zoals in tabel 4.8 is aangegeven wordt de email al meegegeven, mocht de gebruiker de systeememails naar een ander adres willen opsturen dan kan er gebruik van de tweede emailveld gebruikt worden. Na het registreren ontvangt de gebruiker de eerste systeememail waarin wordt bevestigd dat de registratie is gelukt.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Accountgegevens | Gebruikersnaam | URL | Email | Registratie datum |

Tabel 4.8

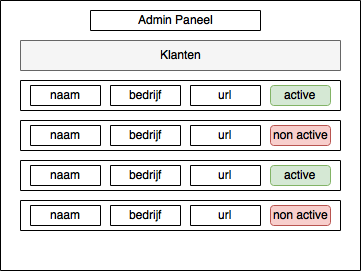
**Authenticatie**

****Het authentiseren van een gebruiker maakt een belangrijk deel uit van het registratie proces. WordPress website gebruiken een eigen database en runnen in hun eigen applicatie omgeving. Het enige wat een gebruikersaccount uniek maakt is de URL op de WWW. Het systeem gebruikt de URL om accounts te identificeren, het probleem hiermee is dat URL overgenomen kunnen worden door andere websites. Dit betekent dat het systeem de klantgegevens moet gaan updaten of verwijderen wanneer dit gebeurt. De klant zal hiervoor contact opnemen met S5 mocht het gebeuren. In Figuur 4.6 wordt er weergeven welke klantengegevens worden opgeslagen. Naast de WordPress accountgegevens en de gegevens die de klant zelf meegeeft is er nog één veld dat door het systeem zelf wordt ingevuld: het active (actieve) veld. Dit is een boolean veld wat betekent dat het Waar (True) of Niet Waar (False) kan zijn. Na het registreren is dit veld initieel False wat met de nummer 0 wordt aangegeven.



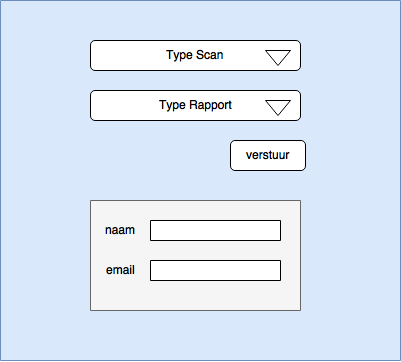
figuur 4.6

**Activatie van account**

Gelijk na het registreren wordt er een nieuw scherm ingeladen, dit is de activatie scherm deze is geïllustreerd in figuur 4.6 en dient als een informatief scherm die aan de klant meedeelt dat de aanmelding in behandeling is. In dit proces zal de administrator valideren of het een authentieke klant is van S5 door bijvoorbeeld contact op te nemen met het bedrijf. Gaat het om een echte klant dan activeert de administrator het account. Wanneer het account geactiveerd is krijgt de klant hier een bevestiging van per email. In figuur 4.7 staat het concept van de admin paneel geïllustreerd. Elke regel is een klant die zich heeft aangemeld via het registratieformulier met een simpele druk op de knop “non actief” wordt het account geactiveerd en veranderd de tekst in “actief”.

figuur 4.7

//verbeter tekst non active

**Aanvraag van scan**

figuur 4.8

Na het activeren van het account is het aan de klant om de extensie te herladen, hiermee wordt bedoeld dat de klant de webbrowser moet herladen om het aanvraagformulier te zien verschijnen. Hoe weet een klant wanneer de browser kan worden herladen? Na het ontvangen van de bevestiging email waarin staat dat het account geactiveerd is, hierin staat ook voor de duidelijkheid dat het scherm herladen moet worden voor het vertonen van het aanvraagformulier. Het formulier bestaat uit vier velden waarvan er twee menu’s zijn en de andere twee tekst velden. De twee menu’s stellen de type scan en type rapport samen. Hieronder twee tabellen met de mogelijke samenstellingen. Er zijn 5 type scans en 2 type rapporten, dit betekent dat er 10 combinatie gemaakt kunnen worden. In hoofdstuk 4.4 wordt er uitgebreid aandacht besteed aan het scan en rapporteer proces. De twee tekstvelden worden gebruikt om een C.C. toe te voegen aan de verzendlijst. Wanneer de klant de aanvraag naar wens heeft samengesteld kan er op de knop “verstuur” gedrukt worden. Het systeem pakt de aanvraag gelijk op en begint met het scannen van de webapplicatie om zo aan het eind van het scan proces een rapport te genereren en deze te versturen naar de klant.

|  |
| --- |
| Type Scan |
| Volledige scan |
| Blind SQLi |
| SQL |
| XSS |
| Snelle scan |

Tabel 4.9

|  |
| --- |
| Type Rapport |
| Volledige rapport |
| Korte Rapport |

Tabel 4.10

## 4.3 API

** API**

Application Programming Interface

Een Application Programming Interface (API) is een verzameling van definities waarmee twee softwareapplicaties met elkaar kunnen communiceren.

Business Doel: Een communicatiemiddel tussen de Client en de Server

De API zorgt voor de communicatie tussen de WordPress extensie en de webapplicatie scanner en is van de type REST API. REST API’s communiceren over HTTP en gebruiken hierbij de HTTP werkwoorden zoals GET en POST.

In het vorige hoofdstuk werden de schermen van de WordPress extensie behandeld, hierin werd de workflow van de extensie uitgelegd. Van het registreren tot het aanvragen van een scan de WordPress extensie roept voor elke handeling die aan de kant van de webapplicatie scanner uitgevoerd moet worden aan via de API, maar ook de admin paneel voert API Calls uit om accounts te activeren en rapporten te verzenden. Hieronder een lijst met alle API Calls.

|  |  |  |
| --- | --- | --- |
| Type | URI | Omschrijving |
| WordPress | HTTP GET /request\_scan | Aanvragen van scan |
| WordPress | HTTP POST /register\_customer | Registreren van klant |
| WordPress | HTTP POST /auth | Het authentiseren van een klant |
| Admin paneel | HTTP GET /admin | Admin paneel opvragen |
| Admin paneel | HTTP GET /customers | Klanten tabel opvragen |
| Admin paneel | HTTP GET /reports | Rapporten tabel opvragen |
| Admin paneel | HTTP GET /active | Activeren van het account |
| Admin paneel | HTTP GET /send{id} | Het verzenden van een rapport |

## 4.4 Webapplicatie scanner

Business Doel: Webapplicatie scanner, Crawler, Scanner, Rapporteren

**Introductie**

In dit deel zal ik inzoomen op het conceptuele ontwerp van de Webapplicatie scanner. Zoals eerder, in de inleiding van dit hoofdstuk vermelde ik wat de kerncomponenten zijn van een Webapplicatie scanner. De database heb ik eruit gelaten omdat het een tot een ander subsysteem behoort.

|  |  |
| --- | --- |
| Type | Categorie |
| Component | Command Line Interface |
| Component | Crawler |
| Component | Scanner |
| Component | Rapporteren |

Tabel 4.8

Commando’s en controle

Een Webapplicatie scanner, zoals iedere software applicatie, heeft een interface nodig die de processen binnen de applicatie start en aansturen. De Webapplicatie scanner heeft als interface een Command Line Interface (CLI). CLI’s bestaan uit een lijst van commando’s die de gebruiker kan uitvoeren een Terminal.

Crawler

De Crawler is een programmeer script dat in verschillende High Level programmeertalen geschreven kan worden. De Crawler doorzoekt de broncode van een website naar specifieke HTML DOM Elementen (Glossary, 11), zoals headers en paragrafen, deze worden gesymboliseerd met <h> en <p> DOM elementen. Na het crawlen wordt er door de ontwikkelaars besloten wat er gedaan wordt met de data.

Scanner

Een van de hoofdfuncties van de Crawler is de Scanner voorzien met data. De Scanner gebruikt de data om gesimuleerde aanvallen te verrichten op het doelwit, de web applicatie. De Scanner voert in de scope van dit project twee aanvallen uit SQLi en XSS aanvallen. SQLi wordt gebruikt om SQL kwetsbaarheden te identificeren en XSS voor cross-site scripting kwetsbaarheden. De Scanner voorziet weer de Rapport generator met data.

Rapporteren

Deze feature genereert een rapport op basis van de gevonden SQLi en XSS kwetsbaarheden. De geïdentificeerde kwetsbaarheden worden op een formele wijze gedocumenteerd in het rapport met gedetailleerde informatie over het doelwit en advies over welke vervolg stappen de klant kan nemen. Het rapport wordt in PDF formaat gegenereerd.

4.4.1 Command Line Interface

Figure 4.8

**Introductie**

De Command Line Interface is het opdrachten prompt van de Webapplicatie scanner. Vanuit daar worden alle opdrachten uitgevoerd zoals de scan opdrachten. De commando’s worden met de web Framework van Laravel gedefinieerd en kunnen in een Terminal uitgevoerd worden. Laravel komt aangeleverd met de CLI Artisan Console, deze CLI ondersteund de Laravel ontwikkelaars bij het ontwikkelen van web applicaties. Laravel heeft een aantal commando’s die ik heb gebruikt bij het ontwikkelen van de Webapplicatie scanner.

Artisan Console

|  |  |  |
| --- | --- | --- |
| id | Artisan commando’s | Omschrijving |
| 1 | php artisan dump-autoload | Genereerd een nieuwe autoload bestand (Glossar,18) |
| 2 | php artisan migrate | Maakt de database aan gebaseerd op de ontworpen schema |
| 3 | php artisan make:migration [tabel] | Maakt een nieuwe tabel aan in de schema |
| 4 | php artisan make:command [commando] | Maakt in het project een nieuwe commando Class aan |

Tabel 4.9

**Custom Artisan commando’s**

Voor het gebruik van de Webapplicatie scanner is het niet nodig om gebruik te maken van deze standaard commando’s die meegeleverd worden met de Artisan Console. Artisan heeft een commando (tabel 4.9, 4) waarmee je een Command Class kunnen genereren. Hiermee kunnen Custom (op maat gemaakt) commando’s gedefinieerd worden.

Voorbeeld van de commando:



Een Artisan commando bestaat uit drie delen:

* De start commando van artisan “php artisan”
* De actie die uitgevoerd wordt “make:command”
* De waarde die de gebruiker aan Artisan meegeeft “CustomCommand”

De commando die in figuur 4.5 wordt getoond genereerd een nieuwe Command Class, dit is tevens ook de basis voor op maat gemaakte commando’s. In Figuur 4.6 toont de gemaakte Command Class. Het bestaat uit twee variabels de **signature** en de **description**. De **signature** definieert de commando, in dit geval: **commando {waarde}** en de description is de omschrijving van de commando. Dit wordt getoond in een lijst van commando’s wanneer de help functie van Artisan wordt gebruikt. Verder heeft de Class nog een handle functie die wordt uitgevoerd wanneer de commando in de CLI wordt aangeroepen. In deze functie moet er een opdracht uitgevoerd worden met de waarde die je meegeeft.

Figure 4.10

**CLI van de Webapplicatie scanner**

De CLI van de Webapplicatie scanner heeft een simpele commando, “scan”. De “scan” commando is voor de Scanner component en heeft een aantal mogelijke opties voor de verschillende scan types.

**Scan commando**

De “scan” commando accepteert één waarde, de URL van het doelwit. Dit start een standaard scan van een web applicatie. De Webapplicatie scanner kan twee type scans uitvoeren een standard scan en een gemodificeerde scan. De standaard scan voert een voor gedefinieerde scan uit. Bij de invoer van de standaard scan commando wordt er op het achtergrond opties ingesteld (Tabel 4.10, Standaard). De gemodificeerde scan, wat een scan opmaat is maakt gebruik van opties om zo een scan na wens te configureren. De configurabele opties worden in het volgende hoofdstuk behandeld. (H 4.4.2)

|  |  |  |
| --- | --- | --- |
| **Type** | **Commando** | **Omschrijving** |
| Standaard | php artisan scan {url} | Er wordt op zowel SQLi en XSS injectie gescand met voor gedefinieerde configuratie |
| Modificeren | php artisan scan {url} {opties} | Er wordt een gemodificeerde scan uit |

Tabel 4.10

**Opties**

|  |  |  |
| --- | --- | --- |
| Optie | Commando (optie) | Omschrijving |
| Blind SQL | --bs | Scan op Blind SQLi |
| SQL | --s | Scan op SQLi |
| XSS | --x | Scan op XSS |
| Headers | --h | Scan op Headers |
| robotTxt | --r | RobotTxt (Glossay, 19) |
| Follow mode | --fm | Volg modus, bepaald welke scope van het domein er wordt gecrawld |
| Report Type | --rt | Volledige rapport of een korte rapport |

Tabel 4.11

**Standaard scan**

Configuratie:

* SQLi : ja
* XSS : ja
* RobotTxt : ja
* Follow mode : 0 – crawl alle URL’s

**Gemodificeerde scan**

****

Configuratie:

* SQLi : ja
* XSS : nee
* RobotTxt : ja
* Follow mode : 2 – crawl alleen URL’s naar de zelfde host.

### 4.4.2 Crawler

**Introductie**

De Crawler behoort tot de kern componenten van een webapplicatie scanner en maakt deel uit van het projectstructuur van het Laravel project (H6). De term “Crawling” is synoniem worden met de activiteit waarbij data van een website programmatisch wordt verkregen. De Crawler voorziet de webapplicatie scanner van informatie over de website, dit maakt deze component een essentieel onderdeel van het gehele systeem. Het is simpel om informatie te verkrijgen over een website, hiervoor dien je geen Crawler te implementeren, een simpele HTTP request is al genoeg om de content van een hele website en daarbij ook de headers te bemachtigen, dit wordt over het algemeen gedaan met een HTTP Client die bijna elke programmeer taal native bezit. Dit is voor een webapplicatie scanner in praktische opzicht niet specifiek genoeg. Webapplicatie scanners verzamelen URL links waar zij specifiek naar zoeken op een web pagina, en niet één web pagina maar liever alle web pagina van een gehele website, zover de Robot.txt1 dat toelaat. De URL’s heeft de webapplicatie nodig om de testen uit te voeren, het liefst ook met de aangekoppelde parameters van een URL erbij. Wat de Crawler vervolgens doet is de data opslaan in een datastructuur opslag, zoals in een database of in cache. Dit maakt het duidelijk dat een simpele HTTP request, wat een HTTP response terug geeft, niet genuanceerd genoeg is om mee te werken. Om de introductie tot de Crawler samen te vatten, de Crawler springt van webpagina naar webpagina binnen de scope van de website, op zoek naar URL’s die het opslaat in een datastructuur van een database of cache oplossing. Dit is een voorbeeld van een simpele Crawler. In de volgende deel zal ik specifiek zijn over wat voor Crawler ik ga bouwen voor de webapplicatie scanner.

Figuur 4.10 – werking van een crawler

**Concept Crawler**

Met de introductie wou ik als doel hebben om de lezer te informeren over wat een crawler is en wat de werkzaamheden zijn ervan. In dit deel zal ik uitgebreid behandelen hoe, de Crawler die ik ga implementeren, in zijn werking gaat. Het doel van de Crawler, zoals eerder vermeld is het verzamelen van informatie dat zich bevindt op een website, in het specifiek URL’s. URL’s worden gebruikten in een latere stadia, bij de uitvoering van gesimuleerde aanvallen.

Ik zal de kern van het systeem in de PHP framework Laravel ontwikkelen, deze maakt gebruik van een package manager Composer, dat ontwikkeld is door Symfony. Waarom is dit belangrijk? Package managers geven ontwikkelaars de mogelijkheid om programmeer libraries te installeren in hen projecten. Twee hiervan gebruik ik voor het ontwikkelen van de Crawler.

Ik heb een eerdere versie van een Crawler zelf proberen te schrijven maar daar was ik niet in geslaagd. Het was beperkt in het verwerken van meta data, dus in dit geval informatie over de gevonden links. Ook kon het maar één pagina crawlen per crawl proces. Deze versie zal ik aan de bijlage van dit document toevoegen.

Na zelf een poging gewaagd te hebben tot het schrijven van een Crawler, heb ik besloten om bestaande Crawl libraries (Glossary, library) te gebruiken voor het ontwikkelen van de Crawler. Dit 1) versnelt het ontwikkelproces en 2) geeft mij toegang tot een groot aantal nuttige functie en configuraties. De twee libraries die ik heb gebruikt voor het ontwikkelen van de Crawler zijn: PHPCrawl en Symfony’s eigen DomCrawler. Dit zijn twee zeer uitgebreide Crawlers die elkaar goed aanvullen bij tekortkomingen. Zowel PHPCrawl en de DomCrawler hebben online documentatie staan waarin de functies van hen API worden behandeld (PHPCrawl), (DomCrawler). Na het bestuderen van de PHPCrawl library kwam ik er snel achter dat een aantal functionaliteiten niet mogelijk waren. Zoals het verzamelen van formulier parameters en het verzenden ervan. Dit is de reden dat ik de DomCrawler heb gekozen om de PHPCrawl library te ondersteunen in het Crawlen van een website. PHPCrawl is voor het verzamelen van URL’s en de meta data ervan zeer geschikt. Uiteindelijk draait het allemaal om het verzamelen van data, dus ik zal in de komende paragrafen het daarover hebben.

**Datastructuur van OWASP ZAP en Acunetix**

Uit het onderzoek dat ik had gedaan over twee DAST-tools heb ik aantal bevindingen gedaan (Onderzoek, 3.3). Ik heb de datastructuren bestudeert om inzicht te krijgen in van welke soort data zij gebruik maken. Deze inzichten heb ik gebruikt om te bepalen wat essentieel is voor het de werking van een webapplicatie scanner.

**Configuratie van PHPCrawl**

PHPCrawl heeft een aantal configurabele opties waarvan er twee interessant zijn voor het Crawlen van een web applicatie:

* setFollowMode – deze instelling heeft vier opties genummerd van 0 tot 4 het heeft betrekking op welke domeinen er worden gecrawld. Om een voorbeeld te geven: de instelling **0** zal alle links crawlen.
* obeyRobotsTxt - instelling voor het volgen van de robot.txt of niet. Twee opties 0 of 1.

**Crawlen naar data**

Het verzamelen van data is het primaire doel van een Crawler maar de vraag luidt: wat voor data? In de context van dit project is het belangrijk dat er zo veel mogelijk nuttige data wordt verzameld. Ik gebruik twee Crawlers die ieder ander soort data verzameld, ik zal in dit deel uitlijnen welke dat zijn. PHPCrawl is in de context van de webapplicatie scanner de hoofd Crawler van website gerelateerde data. PHPCrawl heeft een simpele interface waarmee er met het gebruik van een aantal functies veel data kan worden opgehaald (Glossary, Interface).

**PHPCrawl**

|  |  |
| --- | --- |
| Categorie | Type |
| Data | Algemene informatie |
| Data/Meta Data | Headers |
| Data/Meta Data | Links |
| Data | Parameters |

**Tabel 4.11**

PHPCrawl verzamelt zowel naast hyperlinks ook algemene data over de website, dit vindt de Crawler meestal in de response header die het terug krijgt van de server. Maar de Crawlers hoofddoel is om hyperlink te verzamelen om deze later voor andere doeleindes te gebruiken. Hyperlinks hebben een centrale rol bij het verrichten van webapplicatie scans en het is dus praktisch om zoveel mogelijk meta data te verzamelen over hyperlinks. PHPCrawl beperkt zich wel alleen tot het verzamelen van GET-parameter. De reden hiervoor is omdat PHPCrawl niet gebouwd is om POST parameters eruit te filteren, het zou wel kunnen maar dan zal ik maatwerk moeten toepassen.

Hyperlinks

* URL (Glossary, URL)
* Refererende URL
* Redirect (Glossary, redirect)
* Diepte in de sitemap
* Parameters (GET)

Algemene informatie

* Host (Glossary, host)
* Server
* Status code/protocol
* Content lengte
* Connectie
* Content type

**Filteren**

Een belangrijk aspect van Crawlen naar URL’s is het filteren ervan. Er kunnen een variatie van hyperlinks op een website te vinden zijn. Hyperlinks die buiten de scope vallen, de scope configureerbaar, worden eruit gefilterd. Hyperlink die niet voldoen aan de juiste type/soort van een worden er ook uitgefilterd.

**Voorbeeld**

* Website: www.nieuws.nl
* Scope: domein niveau
* Filter: css, js, jpg, gif

Alle hyperlinks die buiten het domein vallen worden eruit gefilterd en ook de aangegeven bestand types. De volgende links worden eruit gefilterd.

* www.nieuws.nl/style.css
* www.nieuws.nl/afbeelding.jpg
* www.twitter.com/nieuws

Geldige hyperlinks

* www.nieuws.nl/login
* www.nieuws.nl/over-ons
* www.nieuws.nl/binnenland

**Voorbeeld met broncode**

Dit het volgende voorbeeld zal ik illustreren wat PHPCrawl aan data verzameld. Dit zal het concept van de Crawler nog duidelijker maken omdat we precies zien wat de Crawler aan content analyseert.

**Hackableweb**

Voor het voorbeeld gebruik ik mijn eigen gemaakte website, deze noem ik Hackableweb. Hackableweb is een PHP website dat special gemaakt is voor het testen van webapplicatie scanners. Het heet Hackableweb omdat de website makkelijk te hacken is, het voordeel die ik heb bij een zelf ontwikkelde website is dat ik kennis heb over hoe het gebouwd is. Deze website draai ik in mijn eigen lokale omgeving en sinds ik het zelf heb ontwikkeld schend ik geen copyright rechten. Ik heb ervoor gezorgd dat de website kwetsbaar is voor SQLi, XSS en ander soort aanvallen. Zo kan ik meten hoe goed de webapplicatie scanner werkt. Ik zal in de realisatie hoofdstuk test cases maken met gebruik van Hackableweb.

In figuur 4.2 wordt een HTML-pagina geïllustreerd, iets wat de Crawler zou kunnen verwerken. De blauw gemarkeerde tekst zijn de GET en POST URL’s die verzamelt worden, paarse zijn de parameters van de formulieren die bij een POST URL horen en de gele gemarkeerde tekst zijn de bestanden die eruit worden gefilterd.

**Configuratie**

Website: localhost:8888, scope: domein niveau, filter: css, js, jpg, gif



Figure 4.11

**DomCrawler**

Zoals eerder vermeld heb ik de DomCrawler specifiek voor twee taken geïmplementeerd: het crawlen van POST parameters en het verzenden van formulieren. De DomCrawler heeft een geavanceerde filter interface waarmee zeer nauwkeurig op DOM (Glossary, DOM) elementen gefilterd kan worden. Op deze wijze kan ik de input elementen van de form element opsporen. Waarom heb ik de input elementen nodig? De input elementen hebben een attributen “name” waarin de parameters bevinden. In het voorbeeld van figuur 4.3 zijn de twee parameters “username”, “pass” en “login\_submit”. Dit betreft een login formulier van Hackableweb. Voor het verzenden van deze login formulier het de Crawler deze parameter nodig. Dit heeft te maken met het authentiseren proces. Dit zorgt ervoor dat de Crawler zich kan authentiseren indien dat nodig is. Mocht deze niet meegegeven zijn door de gebruiker, dan zal de toegang tot de website geblokkeerd worden. In figuur 4.2 zal ik in het paars aangeven wat de DomCrawler crawlt.

**HTML Form**

Figure 4.12



**Authenticatie**

Als authenticatie vereist wordt van de webapplicatie dan dient de deze door de beheerders (S5) ervan doorgevoerd te worden. De klant hoeft enkel een aanvraag te doen op een scan, omdat het een klant betreft, heeft S5 de benodigde inloggegevens voor het authenticatie proces. Deze zijn voorzien door de klant, zowel de inloggegevens voor de consumenten account en administratie account.

**Sessie management**

Na het inloggen wordt er een sessie ID gegenereerd deze wordt opgeslagen in de sessie tabel zodat er bij elke http request een de sessie ID meegegeven kan worden. Zou behoudt de webapplicatie scanner de sessies die worden aangemaakt.

### 4.4.3 Scanner

#### Introductie

In dit deel zal ik het concept van de Scanner uitgebreid behandelen met als doen op een begrijpelijke manier uit te leggen hoe de Scanner in zijn werking gaat. De Scanner is een fundamentele componenten van de Webapplicatie scanner en maakt deel uit van het Laravel projectstructuur (H6). De Scanner is het component dat de gesimuleerde aanvallen verricht op het gekozen doelwit wat een web applicatie is. Het component heb ik Scanner genoemd omdat het webpagina’s scant op kwetsbaarheden. In de context van dit project zijn de doelwitten webshops van wie de eigenaren klant zijn bij S5. In het deel hiervoor (H 4.4, Crawler) heb ik behandeld wat voor soort data de Crawler verwerkt en gestructureerd opslaat in de database. Deze data wordt door de Scanner gebruikt om de gesimuleerde aanvallen uit te voeren. Voordat ik begin met het behandelen van de soort data dat de Scanner gebruikt zal ik eerst de basis uitleggen.

De scanner zal een modulaire software architectuur hebben zodat het makkelijk uitbreidbaar is met nieuwe modules. Een scanmodule is een aanval types die de Webapplicatie scanner uitvoert. In het onderzoek (H3) ben ik tot de conclusie gekomen dat er in de scope van dit project maar drie aanval types feasible/haalbaar zijn, Blind SQLi, SQLi en XSS. Naast deze 3 scanmodule is er ook een ander type module, de Header module. Deze module voert geen gesimuleerde aanvallen uit omdat het een ander werking heeft, het valt dus niet onder de categorie aanval type. De modulaire software architectuur bestaat uit een basismodule, een blueprint/template, dit is de basis voor elke scan modules. Een andere uitleg is dat de basismodule abstract is, wat betekent dat het verschillende vormen kan aannemen en dat er vanuit de basismodule concrete scanmodules gemaakt kunnen worden. De SQLi en XSS scanmodules zijn dus de concrete scanmodules (Figuur 4.13). De drie aanval modules hebben twee soorten HTTP aanvallen GET en POST. GET aanvallen worden anders opgebouwd dan POST aanvallen.

figuur 4.13

#### Modules

SecurityReport heeft 4 Scan Modules waarmee het kwetsbaarheden identificeert. De Scan Modules (Blind) SQLi en XSS vallen onder hetzelfde categorie “gesimuleerde aanvallen” de Header Module voert geen aanvallen uit maar gebruikt de header data die verzameld is door de Crawler. De drie Modules van de categorie gesimuleerde aanvallen voeren HTTP GET en POST aanvallen uit maar verschillen in de manier hoe de kwetsbaarheden wordt geïdentificeerd. In tabel 4.12 staan de kenmerken van de 4 Scan Modules.

|  |  |  |  |
| --- | --- | --- | --- |
| Scan Module | Identificatie methode | Risico Niveau | Kleur code |
| Blind SQL Injectie | Executie tijd bijhouden | Hoog | Rood |
| SQL Injectie | SQL errors zoeken in Response | Hoog | Rood |
| XSS | Javascript code zoeken in Response | Hoog | Rood |
| Security Headers | Security Headers zoeken | Middel, Laag | Oranje, Geel |

Tabel 4.12

#### Het scanproces

In dit deel worden de verschillende scanprocessen behandeld. De applicatie heeft 2 verschillende scanprocessen die ervoor zorgen dat er kwetsbaarheden worden geïdentificeerd.

|  |  |
| --- | --- |
| Scanprocessen | |
| Type | Omschrijving |
| Gesimuleerde aanvallen | Zoekt naar een bepaalde tekst op een webpagina |
| Gesimuleerde aanvallen (Blind injecties) | Wacht een bepaalde tijd op een reactie van de server |
| Misconfiguraties | Controleert op security headers |

Tabel 4.13

Het scanproces van de gesimuleerde aanvallen bestaan uit 3 stappen:

* Combineerproces
* Aanvalproces
* Analyseproces

Het scannen op misconfiguraties controleert enkel op security headers die niet ingesteld zijn aan de server kant en applicatie klant.

#### Misconfiguraties (Security Headers)

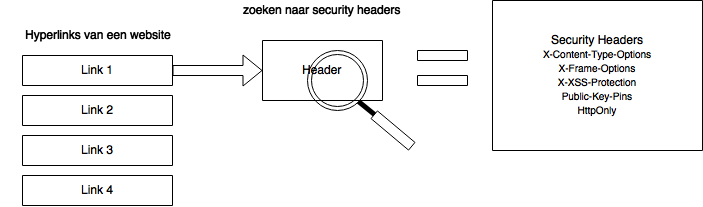
Tijdens het Crawl proces worden er hyperlinks verzameld, deze hyperlinks hebben allemaal een eigen header. Headers bevatten informatie over een webpagina en worden mee verzonden bij zowel een Request en bij een Response. De informatie die in de headers staan zijn vooral regels waar de server (Host) en client (webbrowser) zich aan moeten houden. De security headers zijn ervoor om de server en client te beschermen tegen gebruikers die misbruik willen maken van een server of website.

De Header module controleert op 6 security headers

|  |  |
| --- | --- |
| Security Headers | Omschrijving |
| X-Content-Type-Options | Voorkomt MIME-sniffing van een Response |
| X-Frame-Options | Voorkomt clickjacking in iframes |
| X-XSS-Protection | Voorkomt cross-site-scripting aanvallen |
| Content-Security-Policy | Definieert van welke locatie een bestand ingeladen mag worden. |
| Public-Key-Pins | Voorkomt het verpersoonlijken van gebruikers door aanvallers |
| HttpOnly | Voorkomt het stelen van Cookies |

Tabel 4.13

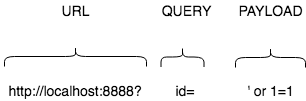
SecurityReport zoekt voor elke hyperlink naar deze 6 security headers wanneer er 1 niet gevonden is wordt het aangegeven als kwetsbaarheid.



figuur 4.14

#### Gesimuleerde aanvallen

**HTTP GET en POST**

Een website heeft over het algemeen twee type URL’s/URI’s. Dit zijn de hyperlinks (GET) die zichtbaar zijn en waarop geklikt kan worden, de ander is onzichtbaar voor het oog en zit verwerkt in een formulier (POST). GET Requests worden via de adresbalk uitgevoerd van een webbrowser. POST Request worden bij het verzenden van een formulier uitgevoerd. Het verschil ligt niet in het feit van waaruit de HTTP Request wordt uitgevoerd maar meer in de opbouw van de HTTP Request.

HTTP GET en POST aanvallen gebruiken URL’s met query’s en payloads waarmee aanvallen worden verricht, en het identificatieproces van kwetsbaarheden is ook soortgelijk. De URL en query’s vormen het eerste deel van de HTTP Request deze staan opgeslagen in de database waar de Scanner via de service laag (H 4.1) toegang tot krijgt. Het tweede deel van de HTTP Request zijn de payloads. De payload is een tekstbestand dat ingeladen wordt in een scanmodule. Payloads verschillen per scanmodule, de scanmodule (Blind) SQLi heeft SQL query’s als payload en de XSS module javascript code. Zoals in de vorige paragraaf aangegeven is er een verschil in hoe HTTP POST de aanval uitvoert ten opzichte van HTTP GET. HTTP POST aanvallen worden toegevoegd aan de Header wat betekent dat de Query in Figuur 4.14 mee wordt gestuurd in Figuur 4.15 staat hiervan een voorbeeld.



figuur 4.16

figuur 4.15

**Combineerproces**

|  |  |
| --- | --- |
| Data | Omschrijving |
| URL | De Unified Resource Locator, het adres waar de aanval op wordt uitgevoerd. |
| Parameters | De parameters die gekoppeld worden aan de URL’s |

Tabel 4.12

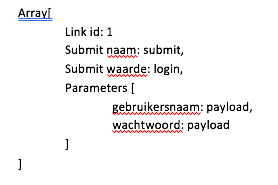
De Scanner gebruikt eigenlijk maar twee waardes voor een scan, de Hyperlinks die geïndexeerd zijn door de Crawler en de parameters die gekoppeld worden met de Hyperlinks in de database, de verbintenis noemt men een Foreign Key relatie (FK). Bij zo’n relatie wordt de Primairy Key identificatienummer “**id**” uit Tabel link, gekoppeld met de “**link id**” in tabel parameter, dat dezelfde identificatienummer representeert als in de Tabel link. Zo weet het softwaresysteem welke parameters bij welke link hoort.

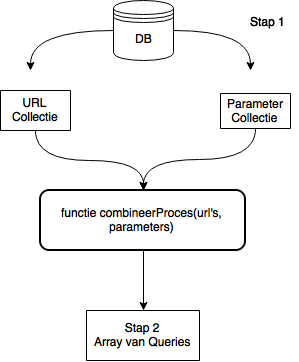


Figure 4.15

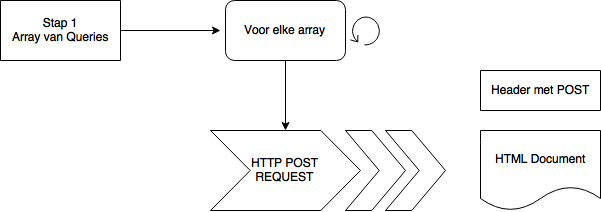
De twee waardes staan in de Database los van elkaar maar worden door het Scanner component aan elkaar gekoppeld. Het koppelen gebeurt wanneer de Scanner in de voorbereidende fase zich klaar maakt voor de gesimuleerde aanvallen. Uiteraard zijn de verschillen er ook bij het combineren van links en parameter. GET heeft een ander combinatieproces dan POST.

In figuur 4.16, 4.17 en 4.18 staat het proces op een simpele manier geïllustreerd. In dit proces worden twee collecties uit de database gehaald de URL’s en Parameters. Uit ieder collectie worden twee items gehaald, deze worden meegegeven aan de functie **combineerProces**, deze combineert de twee waardes aan elkaar om uiteindelijk een samenstelling van URL en Parameter te krijgen. Voor een POST Request eindigt het combineerproces gelijk na de uitvoering van **combineerProces**. De array wordt dan verzonden naar het volgende proces (figuur 4.18) die de HTTP POST Request uitvoert. In figuur 4.17 staat de inhoud van een array.

**Combineren van POST Requests**



figuur 4.17



STAP 2

FIGUUR 4.16

figuur 4.18

De array die in figuur 4.17 getoond wordt bestaat uit vier elementen waarvan parameters een array is.

|  |  |
| --- | --- |
| Link id | 1 |
| Submit naam | submit |
| Submit waarde | Login |
| Parameters | |  |  | | --- | --- | | gebruikersnaam | payload | | wachtwoord | payload | |

Tabel 4.13

In parameters staan de waardes die gebruikt worden bij een aanval. In het geval van figuur 4.17 is het een gebruikersnaam en een wachtwoord wat betekent dat het om een inlogformulier gaat. Om te begrijpen waar de data van tabel 4.14 vandaan komt moet je begrijpen hoe in HTML een formulier gemaakt wordt. Een inlogformulier bestaat uit minimaal twee velden, een veld voor de gebruikersnaam en een veld voor het wachtwoord daarbij hoort er ook een knop bij deze wordt in HTML termen submit genoemd. Een HTTP POST Request gebruikt de velden en de knop om een Request te maken naar de server. Bij het versturen van de Request worden de waardes die gekoppeld zijn aan de velden en de knop verwerkt in een URL zoals in figuur 4.15 wordt geïllustreerd. In het geval van het inlogformulier ziet de URL er zo uit:

gebruikersnaam=payload&wachtwoord=payload&submit=Login

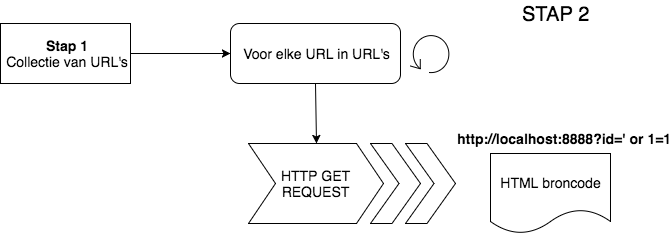
**Combineren van GET Requests**

Bij een HTTP GET Request beëindigd het combineerproces pas na dat alle sub processen zijn uitgevoerd. De sub processen zorgen ervoor dat de GET URL correct wordt opgebouwd.



Tijdens het aanval proces worden alle samengestelde URL’s (stap 1) in geïndexeerde Collectie gezet en klaargezet voor de gesimuleerde HTTP GET aanvallen. Ieder URL van de Collectie wordt uitgevoerd met een HTTP GET Request om een HTTP Response (Reactie) van de server terug te krijgen. In deze Response zit de broncode van de HTML in. Het is de HTTP Response die de Scanner analyseert.

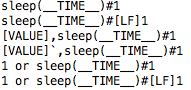
**Formule HTTP = URL X Payload**

De Scanner voert (Aantal HTTP Request = URL’s X Payloads) uit. Als er 10 URL’s zijn en 100 payloads dan worden er 1000 HTTP Request uitgevoerd.

**Analyseproces**

In het analyseproces worden de Responses van de uitgevoerde Request geanalyseerd. Er zijn twee vorm van analyses, een analyse waarbij de tijd van een aanval wordt bijgehouden; dit wordt gedaan bij Blind SQLi aanvallen en een analyse waarbij er gekeken wordt naar de HTML content; dit wordt gedaan bij SQLi en XSS aanvallen.

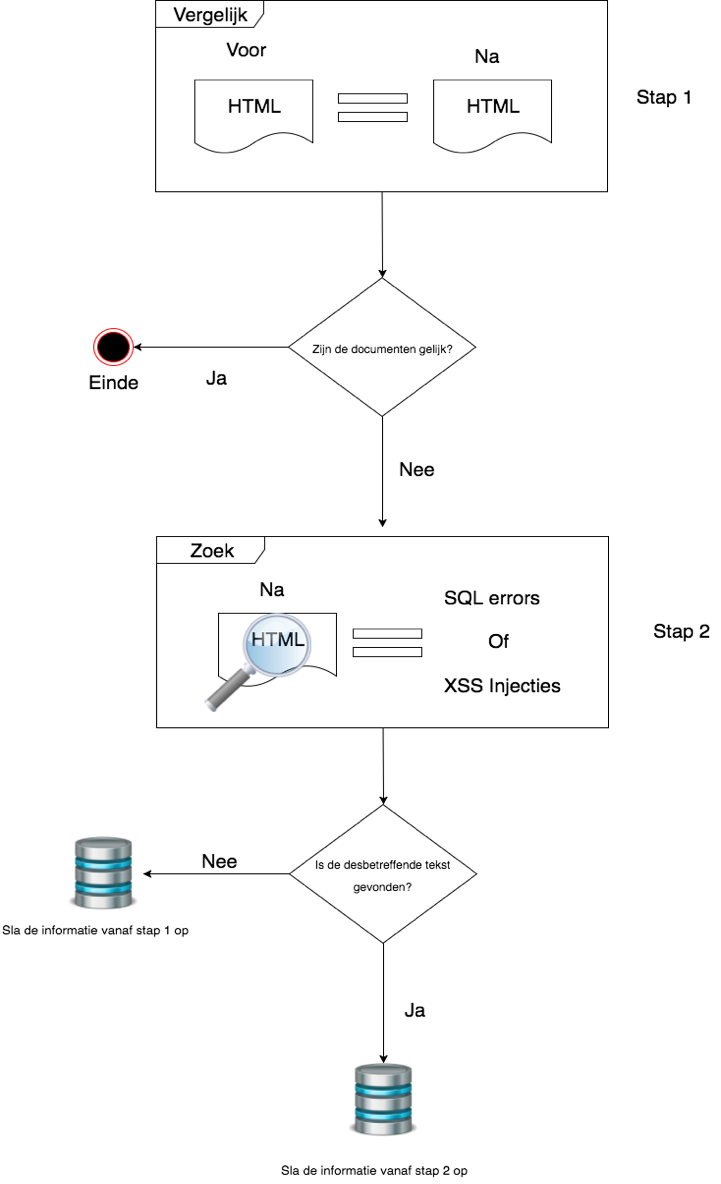
**Analyse Blind SQLi**

Blind SQLi kwetsbaarheden worden ontdekt door de tijdduur van een SQL Query wanneer deze is uitgevoerd. Het doel van Blind SQLi aanvallen is niet om data op te vragen uit een database maar, om te controleren of de server de Blind SQLi uitvoert. Bij de uitvoering van de SQL Query wordt de functie **sleep** gebruikt, deze functie vraagt om 1 parameter wat de tijdsduur is van hoelang de server moet wachten (slapen) voor dat het een reactie (Response) teruggeeft. Bij een succesvolle Blind SQLi aanval voor de server de Query uit en wacht het de zolang als is ingesteld door de hacker, de hacker weet dan dat de server kwetsbaar en gaat dan over naar de wat zwaardere aanvallen, de SQLi aanvallen. In figuur 4.19 woorden een aantal Blind SQLi aanvallen getoond.

figuur 4.19

**Analyse SQLi en XSS**

Het analyseproces voert twee basis handelingen uit, vergelijken en zoeken. Het vergelijkt twee versie van de broncode van een HTML pagina. De versie voor de scan (onaangepast) en de versie na de scan (aanval). Hiermee kan het Scanner component controleren of er de scan effect heeft gehad op de HTML pagina. Mocht dit zo zijn dan is er tekst toegevoegd op de HTML pagina. Dit betekent ook direct dat de web applicatie kwetsbaar is voor SQLi of XSS aanvallen. In figuur 4.20 kun je de flow van het proces zien.



figuur 4.20

**SQLi**

Het controleren van de HTML pagina’s is voor iedere module anders. De SQL module controleert op database errors van meerdere DBMS (Glossary, 12). Hiervoor heb ik een lijst gemaakt met SQL errors die corresponderen met de SQL payload. Met andere woorden als een payload een error veroorzaakt op een HTML pagina dan is het één van deze errors, je kunt het zien als de error scope die ik bepaald heb voor deze scanmodule.



Figure 4.18

**XSS**

Voor de XSS module is er geen voor gedefinieerde lijst met Javascript code regels, omdat je niet te maken hebt met gestandaardiseerde teksten zoals bij SQL die bij een error een bericht toont maar wat Javascript wel kan is script runnen. Met andere woorden er kan Javascript code verstuurd worden naar de server die het dan verwerkt. Zoals er in Hoofdstuk 3.2.2 behandeld wordt kunnen er HTML elementen script “<script>” geplaatst worden op een HTML pagina. Tussen twee van deze elementen kan Javascript code geplaatst worden dat door de server wordt uitgevoerd. De XSS module gebruikt deze feature om zelf opmaat gemaakte XSS script op de HTML pagina van de doelwit te plaatsen. XSS module maakt gebruik van een payload, deze bevat een lijst met XSS aanvallen.

Voorbeelden van de payload



Figure 4.19

Het zijn opmaat gemaakte XSS aanvallen omdat de Scanner component een deel van een payload vervangt e.g. regel 1 (Figuur 4.19). De \_\_XSS\_\_ wordt vervangen door een willekeurige gegenereerde tekst bestaand uit 10 karakters van letters en nummers, e.g. <script>alert(34c829ctn7)</script>.

### 4.4.4 Rapporteren

Business Doel: Het genereren van een rapport op basis van de gevonden kwetsbaarheden

Als uitgangspunt maakt de Webapplicatie scanner een rapport aan gebaseerd op de geïdentificeerde kwetsbaarheden. Het doel van dit rapport is om de klant te informeren over de security staat waar de webshop in verkeerd. Het rapport zal gegenereerd worden in pdf-formaat en zal informatie bevatten over alle gevonden kwetsbaarheden in de webapplicatie. Na dat het rapport is gegenereerd wordt het via e-mail verstuurd naar de desbetreffende klant.

**Inhoud: Welke data bevat het rapport?**

Het rapport heeft als doel de lezer een leesbaar overzicht van alle geïdentificeerde kwetsbaarheden te geven. Naast dat wordt de lezer ook van algemene informatie voorzien en wordt er ook een advies gegeven over de mogelijke vervolg stappen om tot een oplossing van het probleem te komen. Het rapport bevat de volgende data:

|  |  |
| --- | --- |
| Algemene informatie pagina 1 | |
| Veld | Omschrijving |
| Start tijd | Start tijd van de scan |
| Eind tijd | Eind tijd van de scan |
| Scan tijd | De duur van de scan |
| Profiel | Welke type van de scan |
| URL | De URL van het Doelwit |
| Server | De server waarop gehost wordt |

Tabel 4.14

|  |  |
| --- | --- |
| Scan Informatie pagina 1 | |
| Veld | Omschrijving |
| Dreigingsniveau | Niveau van dreiging. Hoog, gemiddeld, laag |
| Totale dreigementen gevonden | e.g. 25 dreigementen gevonden |
| Samenvatting van de dreigementen | Dit bevat een overzicht van elke gevonden dreigement |

Tabel 4.15

|  |  |
| --- | --- |
| Kwetsbaarheden details pagina 2..n | |
| Veld | Omschrijving |
| Kwetsbaarheid | Naam van kwetsbaarheid e.g. SQLi |
| Omschrijving | Omschrijving van kwetsbaarheid |
| URL | Locatie waar kwetsbaarheid is gevonden |
| Aanval | De URI waarmee er een aanval is verricht |
| Impact | Wat voor impact kan het hebben op een webapplicatie |
| Oplossing | Korte oplossing van het probleem |
| WASC ID | Referentie ID |
| Referentie | Hyperlinks die naar bronnen leiden over de kwetsbaarheid |

Tabel 4.16

**Het genereren van het rapport**

Het genereer proces heeft als doel om de data van een gescande website te verwerken in een rapport van pdf-formaat. De Webapplicatie scanner heeft hiervoor een feature dat HTML documenten converteren in pdf-formaten. Dit betekent dat ik de scan data als eerst moet verwerken in een HTML documenten. Een voordeel hiervan is dat de pdf-bestand met HTML en CSS opgemaakt kan worden. Na het converteren wordt het pdf-bestand opgeslagen op de server om vanuit daar opgestuurd te worden naar de klant. Het HTML document wordt zelf ook gegenereerd, hiervoor gebruikt het software systeem een dynamische template. Om de HTML en pdf-documenten te genereren wordt er gebruik gemaakt van twee Laravel Libraries:

|  |  |
| --- | --- |
| Libraries | Omschrijving |
| HTML Builder | Voor het bouwen van dynamische HTML documenten |
| façade | Het renderen van php blade bestanden naar html |
| Html2Pdf | Voor het genereren van pdf-documenten |

Tabel 4.17

In het eerste proces wordt met de template die ontwikkeld is met de HTML Builder een HTML document gemaakt. De data dat de Scanner component heeft opgeslagen wordt verwerkt in de gegenereerde HTML document. In het volgende proces wordt de HTML document geconverteerd naar pdf formaat met Html2Pdf.

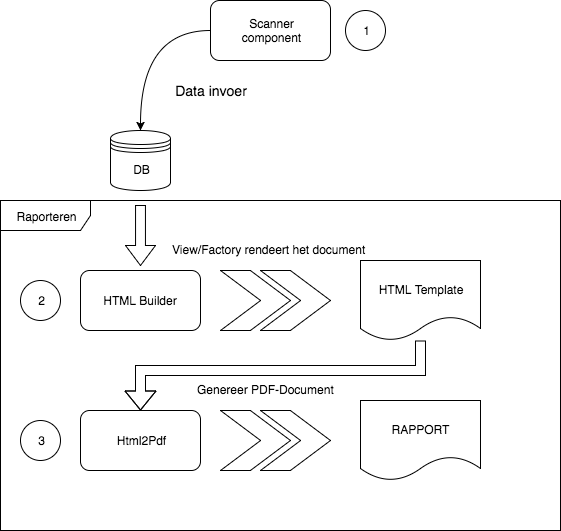
****

Figure 4.20

**Rapport Design**

Het rapport bestaat minimaal uit twee pagina tot n pagina’s. De aantal pagina’s ligt aan de hoeveelheid gevonden kwetsbaarheden. De eerste pagina gebruikt standaard de data uit tabel 4.14 en 4.15. Deze pagina is altijd één groot. De tweede bevat de data uit tabel 4.16, omdat dit een lijst met de geïdentificeerde kwetsbaarheden vertoont kan dit meer dan één paginagroot zijn.

**Eerste pagina**

**Dreiginsniveau**

|  |  |
| --- | --- |
| Totale dreigementen gevonden | |
| Hoog | **2** |
| Gemiddeld | **0** |
| Laag | **0** |

|  |  |  |
| --- | --- | --- |
| Categorie dreigementen | Dreigingsniveau | Aantal |
| SQLi | **Hoog** | **1** |
| XSS | **Hoog** | **1** |

|  |  |
| --- | --- |
| Algemene informatie | |
| Start tijd |  |
| Eind tijd |  |
| Scan tijd |  |
| Profiel |  |
| URL |  |
| Server |  |

Level Hoog

**Andere pagina’s**

|  |  |
| --- | --- |
| Kwetsbaarheden details | |
| Kwetsbaarheid | XSS |
| Omschrijving |  |
| URL | www.webshop.nl?id=<script>alert(34c829ctn7) </script> |
| Aanval | <script>alert(34c829ctn7) </script> |
| Impact |  |
| Oplossing |  |
| WASC ID | 8 |
| Referentie | https://www.owasp.org/index.php/Cross-site\_Scripting\_(XSS) |

**Het verzenden van rapporten**

Business Doel: Het verzenden van het rapport via email

Het verzenden van rapporten en het beheren ervan behoren tot de laatste processen van de Webapplicatie scanner. Het verzendproces is de laatste stap van de service die het softwaresysteem biedt en is het einddoel bereikt. Het business doel van dit deel is: We moeten naast het ontwikkelen van webshop ook meer de focus leggen op security door de klant meer inzicht te bieden op de security staat van hun webshop (H2, business doel). Om dit business doel te bereiken is met het produceren van het rapport het doel nog niet bereikt, dus de laatste stap is dus het verzenden van het rapport naar de klant toe. Voor het beheren van de rapporten refereer ik je naar hoofdstuk 4.4.1.

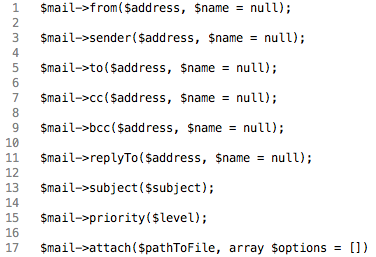
Het is een simpel proces waarbij gebruik wordt gemaakt van een geïntegreerde e-mailservice van Laravel. Laravel heeft een eigen API genaamd “Mail” gebouwd met de Library van SwiftMailer als basis. Het is een simpele API dat de basis functionaliteiten bevat van een mailing service.

Figure 4.21

De vier belangrijkste functionaliteiten zijn

* From
  + Het adres van de verzender
* Sender
  + Het adres van de ontvanger
* Subject
  + Het onderwerp van de email
* Attach
  + Het pdf-bestand dat mee wordt verstuurd (rapport)

Voor de MVP zal ik het houden op deze vier functies. De Attach functie is het meest interessant omdat deze ervoor zorgt dat het rapport in pdf-formaat mee verzonden kan worden. Het vraagt om de bestands locatie van het rapport die opgeslagen staat in een resource map op de server.

Zodra de email ontvangen is door de klant zijn de processen van het Software Systeem afgerond en is het systeem weer bereid om het volgende verzoek te verwerken.

# 5 Eindproduct

Deelvraag

Wat is er gerealiseerd van de Webapplicatie scanner?

**Inleiding**

# Bronnenlijst

**Boeken**

R. S. Sangwan, Software and Systems Architecture in Action

H. Cervantes, R. Kazman, Designing Software Architecture: A Practical Approach

**Internetbronnen**

<https://sucuri.net/website-security/Reports/Sucuri-Website-Hacked-Report-2016Q1.pdf>

<https://assets.documentcloud.org/documents/3527813/IBM-XForce-Index-2017-FINAL.pdf>

<http://essextec.com/wp-content/uploads/2015/09/xforcereport2q2014.pdf>

https://motherboard.vice.com/en\_us/article/vv7mvb/55-healthcare-data-breaches-have-hit-more-than-100-million-people-in-2015

<https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project>

<http://www.sectoolmarket.com/price-and-feature-comparison-of-web-application-scanners-unified-list.html>

<http://pc-en-internet.infonu.nl/geschiedenis/84250-het-web-web-10-web-20-web-30.html>

<https://www.ibm.com/security/data-breach/>

<https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools>

<https://www.acunetix.com/resources/wvsbrochure.pdf>

<https://torquemag.io/2016/10/13-surprising-wordpress-statistics-updated-2016/>

<https://ithemes.com/2017/01/16/wordpress-security-issues/>

<https://www.techempower.com/benchmarks/>

<https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/>

<http://projects.webappsec.org/w/page/13246978/Threat%20Classification>

# Bijlage A

## Glossary

1. **MVP:** Eenminimal viable product is een product dat aan de minimale functionele eisen voldoet. Als voorbeeld neem ik een inlogsysteem. De minimale functionele eisen van een inlogsysteem is dat een gebruiker ermee kan inloggen. Bouw je een “herinner mij” functie in dan is het niet meer minimaal en volgt het niet meer het principe van een MVP.
2. **DAST**: Dynamic Application Security Testing, dit zijn security tools die zich specialiseren in het uitvoeren van Blackbox testing op web applicaties.
3. **Crawl**: Bij een Crawl proces worden HTML pagina’s doorzocht naar specifieke HTML DOM elementen.
4. **CMS**: Een CMS is het systeem achter een website dat het beheren hiervan eenvoudiger maakt. Zo kan iemand met relatief weinig technische kennis een website bouwen en beheren.
5. **API**: Een Application Programming Interface (API) is een verzameling van definities waarmee twee softwareapplicaties met elkaar kunnen communiceren.
6. **URL**: Unified Resource Locator, een adres van website op het internet.
7. **Redirect**: Een gebruiker van een internet browser wordt verzonden van de huidige URL naar een andere URL.
8. **Library**: Een verzameling van code (Classes/functies) die door programmeurs online beschikbaar zijn gesteld e die andere programmeurs kunnen gebruiken bij het ontwikkelen van software applicaties.
9. **Interface**: Gebruikersfunctionaliteiten voor het uitvoeren van programmatische handelingen in een software applicatie.
10. **HTML**:Hyper Text Markup Language, een opmaak taal voor een website.
11. **DOM**: Document Object Model, een HTML pagina is opgebouwd uit DOM elementen zoals de bekende <p> voor paragraaf element.
12. **DBMS**: databasemanagementsystemen, applicatie waarmee SQL databases gemaakt kunnen worden.
13. **Endpoint**: Een Endpoint is een apparaat, software applicatie of netwerk node dat de eindpunt van de communicatie keten is.
14. **Userstories:**
15. **Userpoints**:
16. **CRUD:**
17. **ORM**:
18. **Autoload**:
19. **RobotTxt**: