Afstudeer Verslag
Cloud Security Test Automation

Student:	Niels Loozekoot
Student Nummer:	15081214
Onderwijsinstelling:	De Haagse Hogeschool, Delft
Opleiding:	HBO-ICT: NSE
Afstudeerperiode:	2019-02-04 t/m 2019-05-31
Begeleidend Examinator:	Pieter Burghouwt
Expert Examinator:	Tony Andrioli
Bedrijf:	Secura
Afdeling:	Security Assessment (SA)
Begeleider:	Pim Campers
Opdrachtgever:	Ralph Moonen
Versie:	0.4
Datum:	2019-05-30

VERTROUWELIJK

VOORWOORD

Ter afronding van de HBO-ICT: Network Systems Engineering studie aan de Haagse Hogeschool te Delft is er door Niels Loozekoot een afstudeeropdracht uitgevoerd bij Secura. Het doel van dit document is om te documenteren hoe het afstuderen verlopen is en dat ik in bezit ben over een HBO werk- en denkniveau. Ik zou graag Pieter Burghouwt willen bedanken voor het geduld om meerdere malen feedback te leveren op de documentatie en het helpen met structureren van dit document. Ook Tony Andrioli wil ik bedanken voor het leveren van feedback op het document en het helpen met duidelijk maken waar problemen lagen in het verslag. Voor de hulp met het begrijpen van de opdracht en het geven van ideeën wil ik graag Pim Campers bedanken.

-Niels Loozekoot
2019-05-27

SAMENVATTING

Om de marktdoelen van Secura na te streven is het initiatief naar voren geroepen om een tool te bouwen dat in staat is om automatisch penetratie tests uit te voeren op cloud omgevingen. De tool zal tijdrovende taken wegnemen van werknemers en zal klanten snelle, reguliere en volledige tests aan kunnen bieden die voldoen aan officiële standaarden (baselines) voor security.

Het doel van dit document is om inzicht te geven in het proces dat de stagiair gevolgd heeft om met behulp van Agile dit product tot stand te brengen. Hiervoor is een haalbaarheidsonderzoek uitgevoerd met als onderzoeksvraag: Welke mogelijkheden bieden Cloud diensten aan om pentests te automatiseren en welke informatie is nodig om dit te realiseren?

Om het haalbaarheidsonderzoek uit te voeren is de extensieve documentatie geraadpleegd van Amazon Web Services, Microsoft Azure en Google Cloud. Hierbij is de stagiairs kennis van programmeren en security gebruikt om een lijst van nodige data op te stellen die de cloud diensten aan zouden moeten bieden om een pentest te automatiseren. Met de resultaten van het onderzoek kon een keuze gemaakt worden welke soort tests geautomatiseerd konden worden en welke cloud diensten in staat zouden om pentests te automatiseren.

Op basis van het haalbaarheidsonderzoek en literatuuronderzoek is een programma ontworpen en geprogrammeerd in Python3 dat als framework fungeert. Het programma is in staat om instellingen uit cloud providers te halen en te vergelijken met security richtlijnen (baselines). Aan de hand van de informatie dat het programma ophaalt kunnen rapporten gegenereerd worden. Vervolgens kunnen security experts binnen het bedrijf met deze rapporten penetratie tests / crystal box tests uitvoeren op omgevingen en kan zo tijd bespaard worden. Uiteindelijk kunnen deze test ook zelfstandig uitgevoerd worden zonder dat hier enige interactie van een medewerker voor nodig zou zijn. Als vervolgestap na de stage kan het programma uitgebreid worden met nieuwe functionaliteit dat nieuwe diensten en cloud dienst leveraars ondersteund. Ook kunnen diensten tegen officiële richtlijnen gehouden worden (zoals CIS[3.1]) als de baselines goed ingevuld zijn waardoor het bedrijf industriewijde standaarden ondersteund op het gebied van Cloud Security.

INHOUDSOPGAVE

Bijlage kunnen in het bijlage document gevonden worden.

PERSONEN

Naam	Rol
Niels Loozekoot	Stagiair
Pim Campers	Begeleider vanuit Secura
Ralph Moonen	Opdrachtgever / product owner
Tony Andrioli	Expert Examinator
Pieter Burghouwt	Begeleidend Examinator

VERSIEHISTORIE

Versie	Versiedatum	Controle door	(Officiele) Distributie aan
V0.1	2019-02-20	Niels Loozekoot Begeleidend Examinator	
V0.2	2019-04-30	Niels Loozekoot	Expert Examinator Begeleidend Examinator
V0.3	2019-05-21	Niels Loozekoot Pim Campers Begeleidend Examinator	
V0.4	2019-05-31	Niels Loozekoot Pim Campers	Expert Examinator Begeleidend Examinator Secura

1. INLEIDING

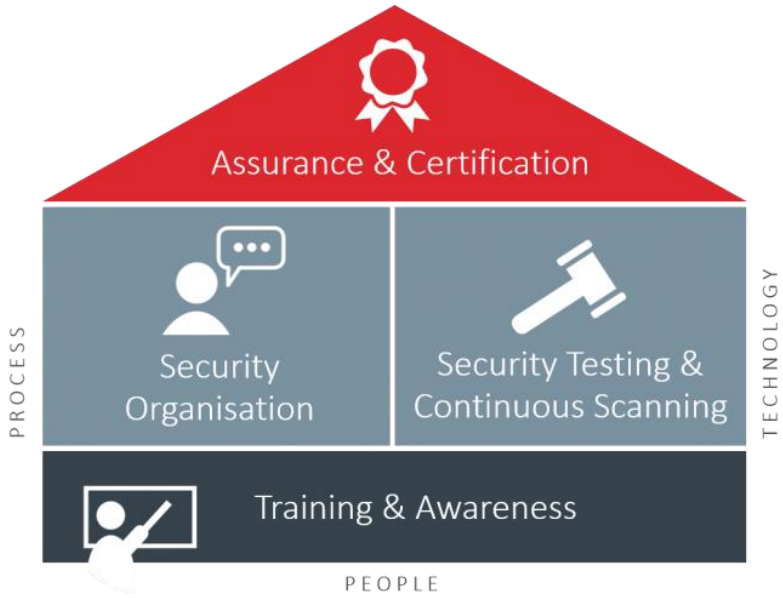
Informatica is een snelgroeiende bedrijfstak en één van de onderdelen van die bedrijfstak veel groei gezien heeft is de Cloud. De cloud wordt steeds vaker gebruikt door bedrijven om hun websites of bestanden in op te slaan, dit geeft aanvallers een extra aanvalsvlak. De stagiair: Niels Loozekoot, is in Februari begonnen aan zijn afstudeerstage bij Secura; een bedrijf met een focus op Security. De stagiair heeft de opdracht op zich genomen om een tool te bouwen dat in staat is om security tests (pentests) in cloud omgevingen te automatiseren. De tool kan veel voordelen bieden aan bedrijven die gebruik maken van cloud diensten.

Met een constante groei in het gebruik van cloud diensten, als dank aan de simpliciteit en flexibiliteit, worden meer bedrijven onderworpen aan de onbekende risico's die cloud diensten met zich meebrengen. Dit naast de risico's die "standaard" servers dagelijks overvallen, zoals recentelijk de nieuwe MDS serie kwetsbaarheden (CVE-2018-12126, CVE-2018-12127, etc.) die alle moderne Intel chips beïnvloed en belangrijke data zoals root wachtwoorden uit het systeem kan lezen. Maar ondanks dat deze diensten al aangeboden worden kosten deze weken of maanden aan tijd om volledig uit te voeren, en zelfs dan kunnen onderdelen ontbreken. Een programma zal deze fout niet maken en zal in staat zijn om alle onderdelen in volledigheid te testen zolang deze geprogrammeerd zijn. Door deze tool te ontwikkelen kunnen bedrijven goedkoper volledige tests ontvangen die meer vinden dan professionals in dezelfde tijd. Dit geeft een securityspecialist de tijd om complexere aanvalstechnieken te onderzoeken.

Om deze tool te realiseren zal gebruik gemaakt worden van de Agile werkmethode om iteratief het programma op te bouwen.

2. ORGANISATIEBESCHRIJVING

Secura, voormalig bekend als Madison Gurkha, is een security bedrijf dat oriënteert op het leveren van Cyber security tests (Red Teaming), Security awareness training, Certificeringen, GDPR/AVG Privacy wet nakoming, en meer. Deze diensten zijn onder te verdelen in het 'huis van Secura', geïllustreerd in de onderstaande afbeelding (2.1). Dit is de focus van Secura.

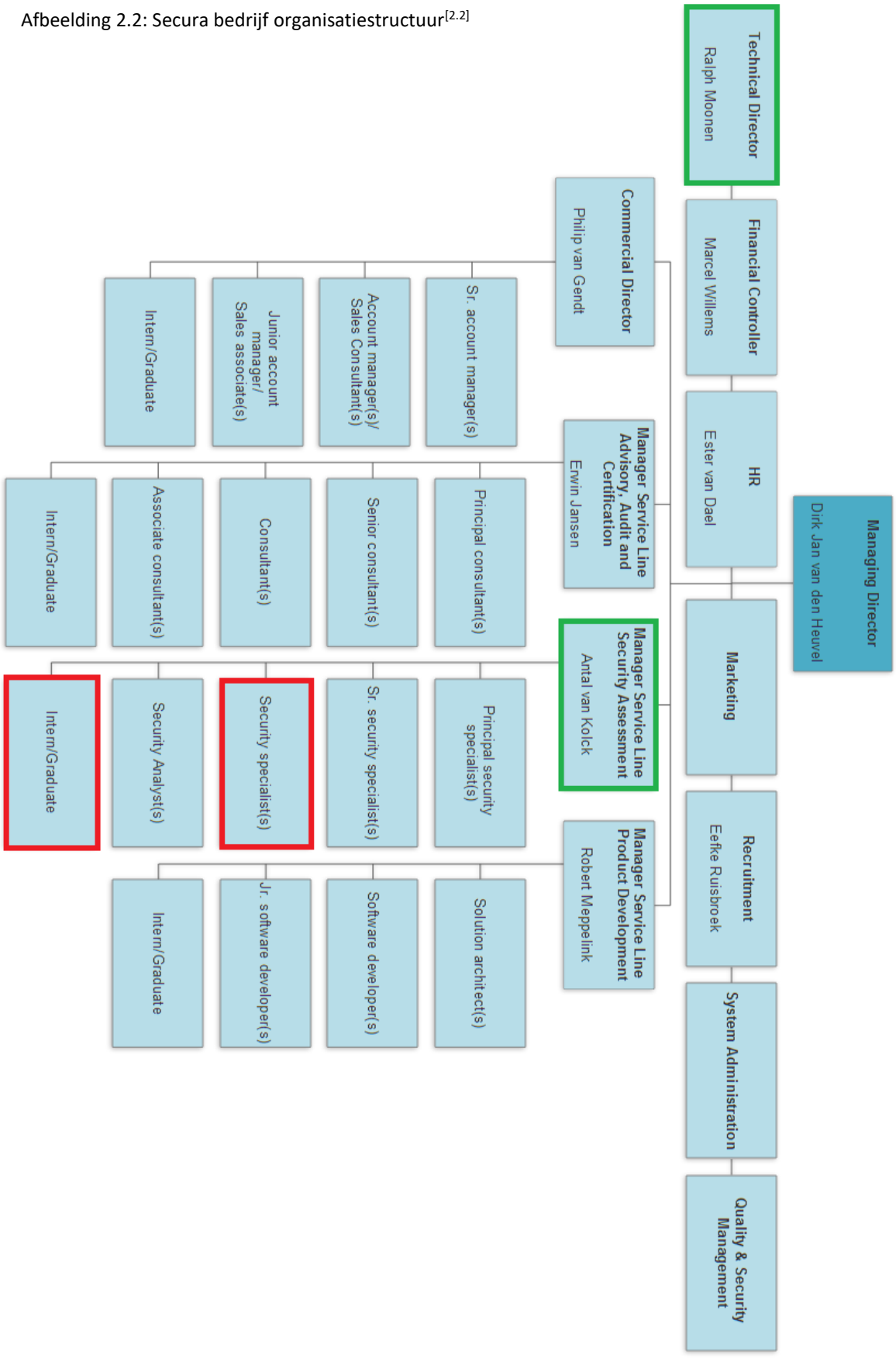


Afb. 2.1 - House of Secura[2.1]

In afbeelding 2.2 is gevisualiseerd hoe de structuur van Secura in elkaar zit. De begeleiders zijn genoteerd met een groene rand, terwijl de stagebegeleider en de stagiair genoteerd zijn in het rood. Secura bevindt zich in een fase van sterke groei en heeft in het afgelopen jaar veel nieuwe werknemers geworven; dit willen ze voortzetten in 2019 en hoopt in 2021 te verdubbelen in aantal medewerkers ten opzichte van 2018, van 70 (nu) naar rond de 120 in 2021. Secura probeert een marktleider te worden op het gebied van security en wilt dit bereiken door bijvoorbeeld hoger op de zoekresultaten te komen bij Google (Zoekterm: "Pentest Nederland"). Ook door goed te presteren bij het vinden van kwetsbaarheden waar andere bedrijven niet in staat waren om iets te vinden en met behulp van mijn product een doorbraak te maken op het gebied van Cloud Security door automatische tests te kunnen bieden voor een goedkope prijs.

Binnen de organisatie is de stagiair in contact met de opdrachtgever, begeleider en een (senior) software developer die de code zal controleren en feedback zal geven. Door de doelen van het bedrijf zijn veel medewerkers (inclusief de hierboven genoemde) erg druk en zijn ook veel 'op locatie' bij klanten. De sfeer binnen het bedrijf kan als 'casual' beschreven worden. Mensen van verschillende posities zitten door elkaar heen, vaak gegroepeerd per afdeling. Het is mogelijk om direct afspraken te maken via outlook. Per de fysieke locaties is Secura in bezit van twee gebouwen. Eén in Amsterdam, met het hoofdkantoor in Eindhoven. De meerderheid van mijn tijd zal in Amsterdam besteed worden in verband met de reistijd tussen huis en werk. Een aantal keer per maand zal ik in Eindhoven zijn voor een gesprek met de begeleider over het project.

Afbeelding 2.2: Secura bedrijf organisatiestructuur[2.2]



3. PROBLEEMSTELLING

Secura wilt een marktleider worden op het gebied van Cloud Security en wil dit bereiken door pentests uit te voeren op de cloud omgevingen van klanten. De pentests die op dit moment uitgevoerd worden door security experts binnen het bedrijf zijn gelimiteerd door tijd en formaat van de omgeving. Bij een bedrijf met honderden systemen is het niet mogelijk om alle invalshoeken af te gaan binnen de korte tijd (vaak <5 dagen).

Bij de pentest krijgt Secura volledige toegang tot de cloud omgeving en de security expert(s) heeft als doel om de omgeving af te speuren op kwetsbaarheden. Deze kwetsbaarheden kunnen zich bevinden in de configuratie van een firewall regel of PaaS instelling en de hoeveelheid regels kunnen bij sommige bedrijven in de honderdtallen liggen. Omdat er geen tools beschikbaar zijn om dit automatisch te analyseren of op te vragen moet de security expert handmatig door alle configuraties klikken om de data op te halen uit van de website. Door de hoeveelheid tijd die dit in beslag neemt wordt er vaak voor gekozen om 10-15 van de instellingen te pakken als voorbeeld en de rest achterwege te laten. Dit zorgt voor een onvolledige test.

Secura wilt deze tests automatiseren zodat:

- Medewerkers meer tijd hebben om aan andere projecten te werken.
- Pentests in volledigheid uitgevoerd worden (niet alleen 10-15 geselecteerde onderdelen).
- Pentests uitgevoerd kunnen worden op virtuele machines op alle configuratie bestanden.
- Alle pentests minimaal aan verschillende richtlijn standaarden kan voldoen. (zoals CIS[3.1])

Een deel van deze functionaliteit zal gebruikt worden tijdens de advertentie van de dienst.

3.1. Doelstelling

Om dit de automatiseren zal de stagiair (Minimaal):
Een alfa versie van de tool/framework ontwikkelen dat:

- Het mogelijk maakt om tests op hele cloud omgevingen uit te voeren zonder de omgeving/layout van tevoren te kennen.
- Alle PaaS en IaaS instanties kan testen, en ten minste de drie cloud providers kan ondersteunen.
- Een systeem / cloud omgeving tegen richtlijnen kan vergelijken.
- Uitgebreid kan worden met nieuwe diensten en baselines – zo simpel mogelijk.

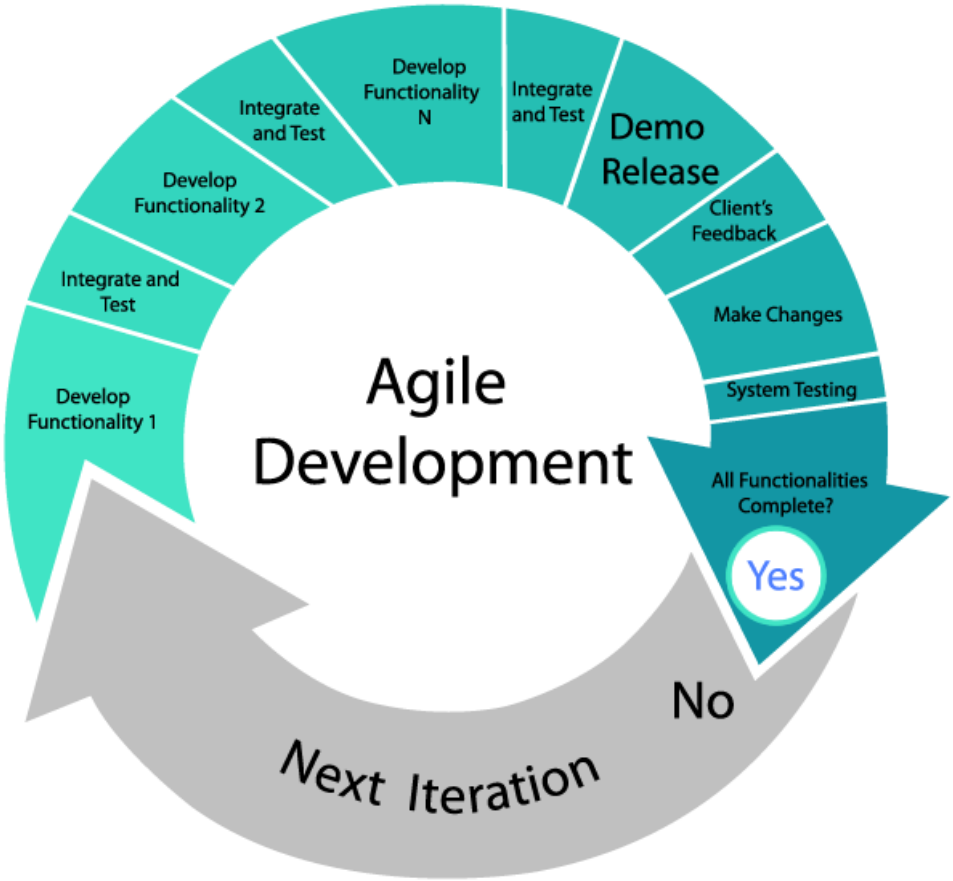
4. AANPAK

Om dit project methodisch en structureel uit te voeren zal het opgedeeld worden in verschillende fases. Deze fases zijn geïllustreerd in tabel 4.1 met daarbij welke producten opgeleverd zullen worden.

Fase	Op te leveren product
Initiatiefase	Plan van aanpak
Definitiefase	Plan van eisen
Onderzoeksfase	Haalbaarheidsonderzoek (Appendix. C) PoC (Haalbaarheidsonderzoek bewijsproduct)
Ontwerpfase	Ontwerpen
Realisatiefase	Software Documentatie
Nazorg	Aanbevelingen & Evaluatie

Tabel 4.1. Fases van afstudeerstage

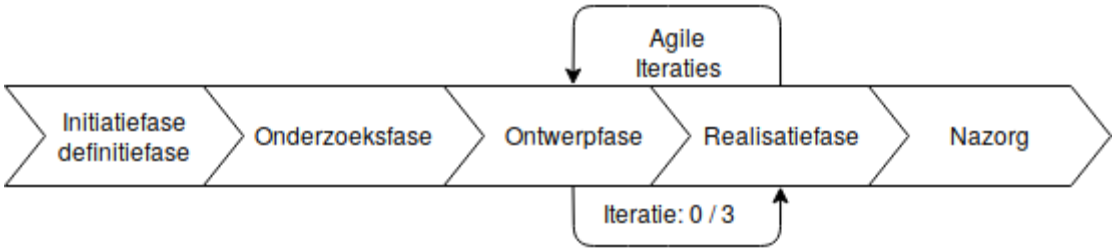
Deze opdeling van fases is gekozen om projectmatig aan de slag te gaan en met behulp van de GOTIK-methode het project in goede banen te leiden. Aan het eind van het haalbaarheidsonderzoek zal een Proof of Concept opgeleverd worden dat aan moet tonen dat de gekozen methodes, implementaties of software/tools haalbaar en implementeerbaar is voor het eindproduct. Deze moet in staat zijn om de kernfunctionaliteit van de tool naar voren te brengen.



Afbeelding 4.2. Agile werkmethode[4.2]

4.1. *Werkmethode*

De werkmethode die toegepast is op het project is Agile (Zie afbeelding 4.2). Het plan van aanpak bespreekt het doorlopen van een V-Model waterval methode in drie iteraties, maar na evaluatie bleek dat het project tegen de grenzen van de Waterval methode aanliep en dat Agile (gebaseerd op het V-Model) een betere en flexibelere optie aanbood die niet veel verandering eiste voor het project. Er is voor deze methode gekozen omdat het project er een is waarbij het van belang is om feedback te krijgen van de product-owner. Deze feedback wordt gegeven aan de hand van een demo die ongeveer één keer per vier weken plaats vindt. Ook geeft Agile een gestructureerde omgeving om aan het project te werken met ontwerp en testfases. De Agile development werkmethode zal drie keer doorlopen worden. Deze iteraties zullen afgerond worden met een Demo (samen met de opdrachtgever) en een feedback / een review sessie met de stagebegeleider (van Secura). Elke iteratie zal de 'ontwerp' en 'realisatie' fase van het project opnieuw doorlopen (Zie afbeelding 4.3). Deze afbeelding is opgesteld om de fases van het project op een overzichtelijke manier te volgen door het afstudeerverslag heen.



Afbeelding 4.3: Fasering illustratie

De drie 'iteraties' zijn te benoemen als:

- Proof of Concept
- Prototype
- Prototype (2e iteratie)

Initiatiefase

In de eerste fase van het project zullen verschillende gesprekken gevoerd worden met de begeleider en opdrachtgever om een beter beeld te krijgen van de opdracht. Een plan van aanpak wordt gemaakt met de ideeën en planningen van het project.

Definitiefase

Tijdens de definitiefase worden gesprekken gehouden met verschillende medewerkers en begeleiders met als doel om informatie op te doen over het probleem, wat de visie is van de gebruikers, in te lezen in de opdracht en samen met de opdrachtgever een lijst van eisen op te stellen. Tijdens deze fase wordt ook het haalbaarheidsonderzoek uitgevoerd dat als doel heeft om op te stellen of de gekozen invalshoek/aanpak van de opdracht haalbaar en realistisch is. Hiermee wordt voorkomen dat er open een later stadium in het project erachter gekomen wordt dat de gekozen aanpak niet kan leiden tot een succesvolle afronding van het project. De onderzoeksvraag hiervoor is: "Welke mogelijkheden bieden Cloud diensten aan om pentests te automatiseren en welke informatie is nodig om dit te realiseren?"

Onderzoeksfase

In de onderzoeksfase wordt onderzoek gedaan naar de opdracht en wat nodig kan zijn om de opdracht af te ronden. Literatuuronderzoek wordt gedaan om informatie op te doen over de mogelijke aanpakken en software of framework's om dit project uit te voeren. Aan de hand van het inlezen in het project en onderzoeken van onderwerpen kunnen de nodige producten ontwikkeld worden.

Iteratie 1: (Proof of Concept)

Het proof of concept heeft als doel om na het onderzoek, te bewijzen dat de bevindingen van het onderzoek toepasbaar zijn en dit de demonstreren voor de opdrachtgever. Het hoeft niet aan alle eisen te voldoen omdat het doel van het product is om een (theoretische) mogelijkheid te tonen. In plaats daarvan zal het proof of concept focussen op het vervullen van één of twee onderdelen die theoretisch getest kunnen worden bij een klant in de cloud omgeving. Dit kan in een testomgeving gecontroleerd worden. Wel wordt het proof of concept ontwikkeld op een manier waar het prototype makkelijker mee verder kan werken.

Iteratie 2: (Prototype)

Het prototype zal proberen om een groot deel van de 'must have' eisen of kernfunctionaliteit te implementeren. Dit product geeft de opdrachtgever een duidelijk beeld van hoe het eruit gaat zien en welke functionaliteit terug zal komen in het eindproduct. Indien de opdrachtgever opmerkingen heeft over efficiëntie, ontwerp, lay-out of functionaliteit kan hij deze delen zonder dat dit de 'workflow' van de opdracht beïnvloed, mede te danken aan Agile. De feedback zal meegenomen worden tijdens de volgende iteratie van het project.

Iteratie 3: (Prototype)

De tweede iteratie van het prototype heeft als doel om te voldoen aan de meest belangrijke eisen die door de opdrachtgever gesteld zijn. Dit houdt in dat het – theoretisch – een framework moet zijn dat (grotendeels) implementeerbaar is in een productieomgeving als een vroeg alfa product. Dit is tevens het laatste softwareproduct dat opgeleverd wordt aan het stagebedrijf.

4.2. Beheersing

Om te verzekeren dat het project zonder verdere problemen verloopt zal er tijdens verschillende fases, met behulp van de GOTIK-methode het project beheerst worden. Deze methode maakt gebruik van vijf onderdelen die tijdens het project gecontroleerd en beheerst moeten worden. Deze keuze is gemaakt omdat elk project omgaat met deze onderdelen in verschillende mate, en door tijdens mijn project rekening te houden met deze onderdelen en voorbereidingen te treffen voor mogelijke problemen wordt het project in goede banen geleid.

De stageopdracht heeft het meeste belang bij de beheersing van: Geld, Informatie en Kwaliteit.

Geld

Voor de test- en ontwikkelomgeving zullen verschillende cloud diensten gekocht/gehuurd moeten worden & deze kosten geld. Met behulp van verschillende 'gratis' abonnementen kunnen de kosten significant ingeperkt worden. Het is daarom de verwachting dat er tijdens dit project weinig kosten gemaakt zullen worden. In verband met de kosten is het belangrijk dat er een akkoord gesloten wordt met Secura om de gemaakte kosten te vergoeden. Zonder deze aanvraag is er geen geld beschikbaar, en daarom is dit belangrijk om te beheersen.

Organisatie

Dit project zal door alleen de stagiair uitgevoerd worden; er wordt niet samengewerkt met andere medewerkers (in team verband). Wel is het van belang dat er – in verband met de drukte – op tijd afspraken gemaakt worden met de begeleider on opdrachtgever die wel invloed hebben op het product.

Tijd

Omdat de stagiair werkt zonder samenwerking van andere medewerkers hoeft er geen rekening gehouden worden met het beheren van de tijd van andere medewerkers. De stagiair moet werken binnen een deadline en planning (H4.3) en moet zijn tijd beheersen om dit te halen. Om ervoor te zorgen dat het project niet onderschat wordt, word er een analyse uitgevoerd op het project om de omvang goed in kaart te brengen.

Informatie

Voor de looptijd van het project moet goed gecommuniceerd worden over de opdracht met de begeleider en opdrachtgever om de beste feedback te ontvangen. De gevonden kennis moet beschreven en gedocumenteerd worden zodat het bedrijf daarmee kan werken of ernaar kan refereren op een later moment, na de stage. De enige belanghebbende waarmee informatie gedeeld moet worden zijn als volgt: De opdrachtgever, de (stage)begeleider en de code-reviewer (Zie kwaliteit).

Kwaliteit

Omdat de code verder onderhouden moet worden door het bedrijf na de afronding van de stage is het van belang om de kwaliteit van het product te beheersen. Om de kwaliteit van het product te beheersen zullen er product reviews gedaan worden met de product owner en begeleider. Dit verzekert dat de kwaliteit van het product overeenkomt met de eisen van het bedrijf.

Om de kwaliteit te beheersen van de code zullen code reviews gedaan worden door een medior/senior developer van het product development team. Dit verzekert dat het product voldoet aan de kwaliteitseisen van het bedrijf. De aangewezen persoon hiervoor is de product development team leader, Robert Meppelink.

Overige Risico's

In dit sub-hoofdstuk worden risico's behandeld die niet direct beheerst worden door de GOTIK-methode. Deze worden uitgewerkt met informatie over het risico en wat ondernomen (kan) worden om deze te beheersen of inperken.

Toevoegingen aan de Eisen

Een risico dat voor kan komen tijdens het project is dat de opdrachtgever komt met nieuwe eisen waar nog geen rekening mee gehouden was. Om de kans hierop te verminderen worden gesprekken gevoerd met de opdrachthouder en begeleider om de eisen van het product vast te stellen. Door deze gesprekken te voeren wordt de kans verminderd dat er een eis is die niet opgenomen is in de lijst.

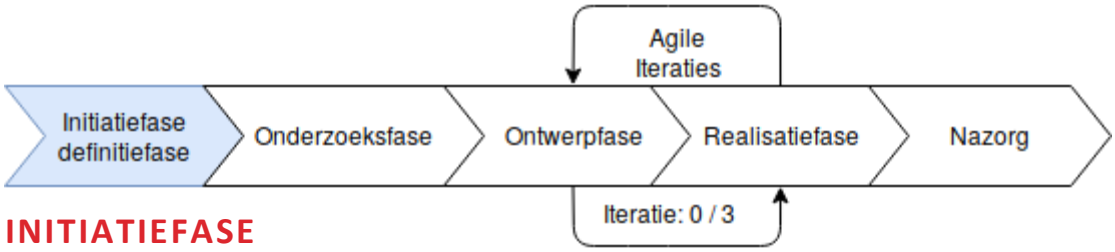
4.3. Planning

Als voorbereiding op de opdracht heeft de stagiair een planning gemaakt van de verschillende fases die doorlopen worden om het eindproduct op te leveren. Hieronder is een tabel weergegeven met de verschillende weken waarin producten opgeleverd worden.

Datum	Op te leveren product	Omschrijving
#1 Ma 4 Feb – Vr 8 Feb	Plan van Aanpak	Initiatiefase
#2 Ma 11 Feb – 15 Feb	N.v.t.	Meelopen en onderzoeken hoe het proces binnen het bedrijf verloopt.
#3 Ma 18 Feb – Vr 1 Maart	Haalbaarheidsonderzoek Onderzoek documenten	Definitiefase Onderzoeksfase
#4 Ma 4 Maart – Vr 15 Maart	Ontwerpen Proof of Concept	Iteratie 1 (Proof of Concept)
#5 Ma 18 Maart – Wo 17 April	Ontwerpen Prototype	Iteratie 2 (Prototype)
#6 Do 18 April – Vr 24 Mei	Ontwerpen Prototype Documentatie	Iteratie 3 (Prototype)
#7 Ma 27 Mei – Vr 31 Mei		Nazorg

Tabel 4.4: Planning

Het eindproduct zou in aan de haalbare wensen van de opdrachtgever moeten voldoen omdat deze twee mogelijkheden gehad heeft om feedback op het product te geven. Dit minimaliseert de kans dat de eisen verkeerd geïnterpreteerd zijn en dat het eindproduct (bijna) direct opgenomen kan worden.

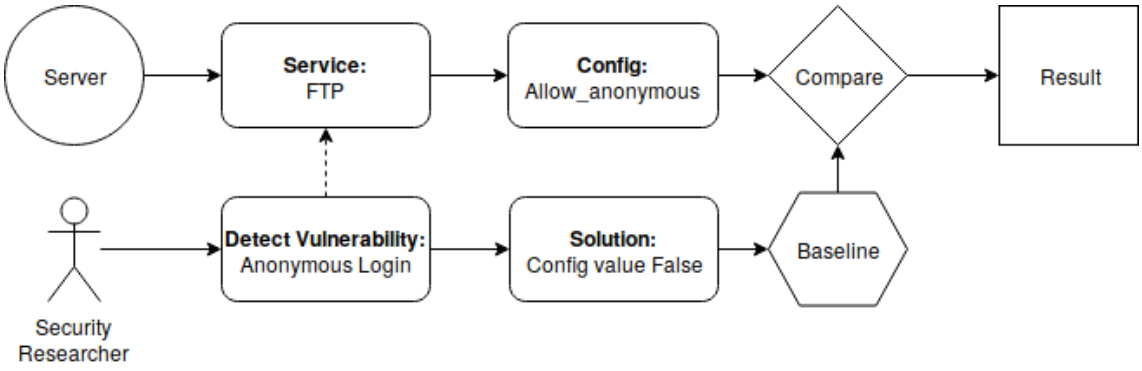


5. INITIATIEFASE

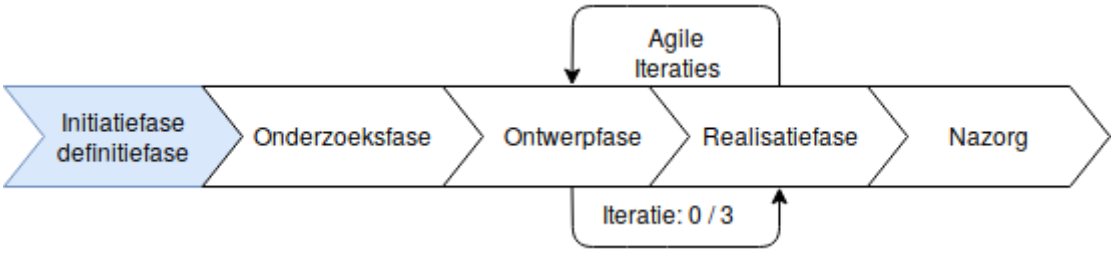
In de initiële fase van de stage zij verschillende gesprekken gepland met de begeleider Pim Campers om te verduidelijken wat de opdracht was en hoe deze aangepakt zou worden.

Uit de gesprekken is gekomen dat het programma in staat moet zijn om de configuraties van de systemen of cloud omgevingen te controleren op richtlijnen (baselines). Dit heeft als doel om configuratie kwetsbaarheden te voorkomen. Eén voorbeeld hiervan is dat FTP-software de mogelijkheid biedt om in te loggen als 'guest' of 'anonymous'. Dit wordt toegestaan door de standaardconfiguratie regel: "allow_anonymous=yes" maar is in bijna alle bedrijfsscenario's niet gewenst omdat het bij bedrijven vaak gaat om servers met privé documenten van het bedrijf of klanten. Dus de baseline (richtlijn) voor deze FTP-software zou zijn dat allow_anonymous op "no" staat. Als dit niet het geval is voldoet het systeem niet aan deze baseline. In afbeelding 5.1 is een visualisatie gemaakt waarbij een Security researcher een kwetsbaarheid vindt en daarvoor een oplossing bedenkt. Hieronder is de bijbehorende uitleg uit het onderzoek:

"In this image a Server hosts an FTP-service. A security researcher discovers that the default configuration of that FTP service allows for anonymous logins. The way to resolve this problem is by setting a value on the configuration to false. If the configuration entry name is "Allow_anonymous" our baseline could be: "FTP Service X requires Allow_anonymous to be false". By comparing the current value of the config to the baseline we can determine if the server adheres to this baseline or not."[5.1]



Afbeelding 5.1: Baseline comparison visualization.[5.1]



6. DEFINITIEFASE

Tijdens deze fase zijn meerdere gesprekken met de begeleider van Secura ingepland die ook één van de gebruikers zal worden om te bespreken wat hij weet over het project, wat zijn visie is over de aanpak en wat hij graag terug zou willen zien in het eindproduct. Daartussen is één gesprek ingepland met de opdrachtgever (Ralph Moonen) om te bespreken wat de eisen zijn van het project en hoe hij graag zou willen zien dat deze tot stand komen.

De eisen die opgedaan zijn uit de gesprekken zijn onderverdeeld in drie verschillende categorieën met als doel om prioriteiten vast te stellen voor tijdens dit project. (Kleine aanpassingen die tijdens het traject gemaakt zijn, zijn ook in deze lijst aangepast).

Must Have

Deze lijst bevat de verschillende functionaliteit die de opdracht 'moet' bevatten volgens de opdrachtgever.

1. Kan een lijst van systemen en diensten produceren binnen minimaal één cloud omgeving.
2. Kan PaaS configuraties ophalen uit de cloud.
3. Kan netwerk configuratie instellingen ophalen uit de cloud.
4. Kan logging configuratie instellingen ophalen uit de cloud.
5. Kan de software detecteren die draait op een IaaS (Apache of Nginx).
6. Kan modulair nieuwe baselines, software en operatie systemen toevoegen.
7. Kan baselines definiëren aan de hand van OS, provider of dienst.
8. Kan de waardes van configuraties / bestanden vergelijken met een baseline.
9. Kan bewijsvoering per baseline genereren (kort).
10. Kan de output van baselines controles naar de reporting tool sturen met LaTeX.

Should Have

De lijst die hieronder beschreven staan bevat onderdelen die de opdrachtgever graag zou willen zien indien de mogelijkheid zich voorstelt.

11. Kan een test via cronjobs uitvoeren.
12. Kan detecteren wat het doel is van (IaaS) servers (Mailserver, Website, etc.).
13. Kan PaaS diensten controleren op configuratie fouten met baselines.
14. Kan netwerk configuratie testen met baselines.
15. Kan logging configuratie testen met baselines.
16. Kan ACL's en firewalls configuratie testen met baselines.

Could Have

In de onderstaande lijst zijn de eisen beschreven die in de tool verwerkt kunnen worden indien de tijd het toelaat.

17. Kan uitgebreide bewijsvoering tonen voor baselines (t.b.v. rapportage).
18. Kan controleren of interactie tussen services via SSL loopt.

6.1. Invalshoek

De reden dat de onderzoeken uitgevoerd worden is om (nieuwe) mogelijkheden te creëren om het project uit te voeren en te zorgen dat alle mogelijke invalshoeken goed onderzocht en overwogen worden. Tijdens het haalbaarheidsonderzoek wordt uitgezocht welke methodes beschikbaar zijn om de websites en informatie te benaderen. Mocht er – bijvoorbeeld – een API beschikbaar zijn, dan is het relatief simpel om informatie daarvan op te vragen. Zonder API wordt het significant lastiger om informatie uit het systeem op te halen. Er moet dan gekeken worden of het mogelijk is om met behulp van een browser de acties van een gebruiker na te bootsen en vervolgens van pagina naar pagina te bewegen om alle informatie daarvan op te halen. Het is zelfs mogelijk dat er een andere mogelijkheid is die nog niet vaak toegepast wordt in het bedrijfsleven. Ondanks dat het ophalen van informatie via de website als 'nepgebruiker' suboptimaal is, is het een geldige aanpak die altijd mogelijk is zolang de opdracht/test ook handmatig uitgevoerd kan worden. Mocht er een API bestaan maar verschillende onderdelen missen, dan zijn er misschien al andere tools beschikbaar die deze informatie wel op kunnen halen of misschien moet er een hybride versie komen dat zowel de website als API raadpleegt. Het haalbaarheidsonderzoek geeft ons inzicht in de mogelijkheden en hoe realistisch deze mogelijkheden zijn. Gebaseerd op voorkennis wordt verwacht dat API als invalshoek gekozen wordt.

6.2. Haalbaarheidsonderzoek samenvatting

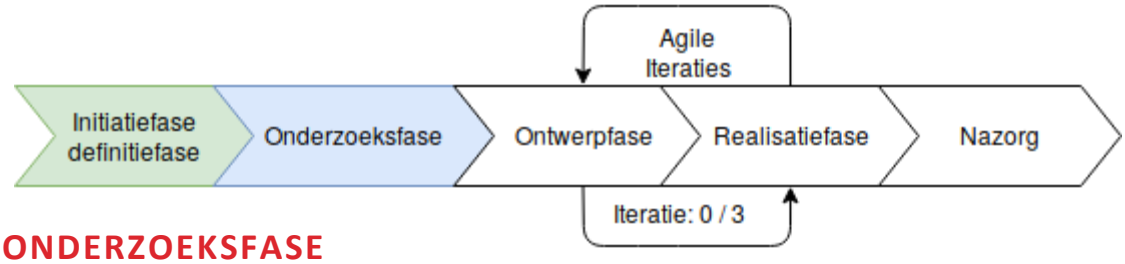
Tijdens dit onderzoek is de vraag gesteld: Welke mogelijkheden bieden Clouddiensten aan om pentests te automatiseren en welke informatie is nodig om dit te realiseren? Om dit onderzoek af te kaderen is er een limiet gelegd op de drie populairste Cloud Serviceproviders: Amazon AWS, Microsoft Azure, en Google Cloud. Deze zijn gekozen op basis van de gesprekken met de opdrachtgever en begeleider. Tijdens dit gesprek is naar voren gekomen dat dit de meest voorkomende cloud omgevingen zijn voor Secura. Dit komt overeen met de drie populairste cloud providers volgens een onderzoek door Computable genaamd "Cloud Survey: Dutch Skies".

Het doel van dit onderzoek is om te controleren of de functionaliteit die de CloudProvider aanbieden om informatie op te halen uit de cloud in staat zijn om nodige informatie te leveren om de tool te bouwen en de nodige tests uit te voeren (zoals een REST API). Vervolgens is er in het onderzoek een lijst van eisen opgesteld over de beschikbaarheid van benodigde informatie die nodig waren van de cloud providers om een pentest te automatiseren (zie Tabel 6.1). Deze lijst was gebouwd op basis van de eisen en persoonlijk ervaring met zowel pentesting als ontwikkeling en de kennis van de projecteisen. De drie populairste cloud providers zijn toen ieder onderzocht op manieren om informatie van de systemen op te halen. Om dat te onderzoeken is de documentatie geraadpleegd van iedere provider en zijn de resultaten van bevindingen uitgetest om te controleren of het werkte zoals verwacht. Het bleek bij het onderzoek alleen mogelijk te zijn om de service te beïnvloeden door middel van de officiële website en een (REST) API. Ook was er een Command line tool beschikbaar voor alle drie de omgevingen die interactie met de API makkelijker maakte om te testen of elke cloud provider in staat was om te voldoen aan de eisen.

Deze eisen zijn vervolgens getest door in een testomgeving de drie command line tools te gebruiken om elke eis van informatie/data op te vragen. Uit de resultaten van het onderzoek (Tabel 6.1) is te zien hoe de drie cloud providers voldoen aan de gestelde eisen en of deze dus bruikbaar zijn voor het automatiseren van pentests. De tabel is vertaald naar het Nederlands vanuit het Engels. Het is hieruit op te maken dat het gebruik van logboeken bij twee van de drie extra geld kost.

Data Eisen	Amazon Web Services	Microsoft Azure	Google Cloud
Server Lijst	REST API	REST API	REST API
PaaS Configuratie	REST API	REST API	REST API
Security Groep lijst/regels	REST API	REST API	REST API
Gebruiker/Groepen: info	REST API	REST API	REST API
Logboeken inzien	REST API, Betaald	REST API, Betaald	REST API
Commando's uitvoeren (IaaS)	REST API & SSH	REST API & SSH	SSH
Geschikt	**Ja**	**Ja**	**Ja**

Tabel 6.1. Vergelijking tussen CloudProviders ten opzichte van de eisen in het onderzoek.



7. ONDERZOEKSFASE

Om in te lezen in de omgeving waarmee de stagiair ging werken zijn er verschillende literatuurwerken geraadpleegd. Deze literatuurwerken zijn opgezocht via Google met als doel om meer te leren over beveiliging en beheer in cloud omgevingen er is daarom veel gelezen over hoe Microsoft Azure, Amazon AWS en Google Cloud veelal beheerd worden, hoe bedrijfsnetwerken opgezet kunnen worden binnen de cloud, hoe de API's van de cloud serviceproviders functioneren of welke capabiliteit ze bezitten. Deze kennis kan vervolgens toegepast worden in het project om diensten beter te kunnen testen op kwetsbaarheden of ervoor te zorgen dat de omgeving goed geconfigureerd is.

Hieronder zijn een aantal literatuurwerken die geraadpleegd zijn tijdens het literatuuronderzoek:

- Amazon AWS Security best practices[7.1]
- Amazon IAM best practices[7.2]
- Amazon AWS Python SDK Documentation[7.3]
- Amazon Boto3 Documentation[7.4]
- Kenneth Reitz (2016). "Structuring Your Project"[7.5]
- Azure API SDK documentation[7.6]

Alle documenten die geraadpleegd zijn, inclusief de informatie die eruit opgenomen is, zijn ook terug te vinden in het rapport van het onderzoek. (Appendix: C. Research Automated Pentesting)

7.1. Onderzoek: Product selectie / aanpak

Om een keuze te maken waarmee het automatische cloud pentest platform gerealiseerd zal worden is er een ondersteunend onderzoek uitgevoerd met als doel om een keuze te maken tussen verschillende softwarepakketten of het zelf programmeren van een programma. In dit onderzoek is er een vergelijking gemaakt tussen drie verschillende programma's, er is hierbij een vergelijking gemaakt met de eisen en extra elementen die belangrijk kunnen zijn zoals of het open source is. De drie softwarepakketten waarmee een vergelijking is opgesteld zijn:

- Ansible
- Chef
- Rudder

Deze drie zijn gekozen op basis van verschillende zoekresultaten naar 'automated cloud testing tools' en vergelijkbare. Deze drie kwamen hierbij het meest naar voren als potentieel geschikte kandidaten met veel bestaande gebruikers. (Meer gebruikers -> betere support/actievere community).

In tabel 7.1 zijn de resultaten van het onderzoek gevisualiseerd op basis van de eisen die gesteld zijn. Hieronder is een lijst van de eisen en de nummers die hierbij horen. De eisen en tabel zijn vertaald vanuit het Engels.

1. Kan systemen / diensten detecteren in onbekende omgevingen
2. Kan detecteren welke soort software draait
3. Kan baselines/richtlijnen controleren afhankelijk van attributen (i.e. versie of OS)
4. Kan integreren met een of meerdere cloud providers
5. Kan integreren met nieuwe PaaS instanties (die nog niet standaard geïmplementeerd zijn)
6. Kan instellingen vergelijken met richtlijnen (en richtlijnen aanmaken)
7. Kan resultaten naar een rapport of API duwen/sturen.

Requirements	Ansible	Chef	Rudder	Custom
1	✓	X	X	✓
2	✓	✓	X	✓
3	X	X	X	✓
4	✓	✓	X	✓
5	X	X	X	✓
6	✓	✓	✓	✓
7	✓	✓	X	✓
Geschikt	**Nee**	**Nee**	**Nee**	**Ja**

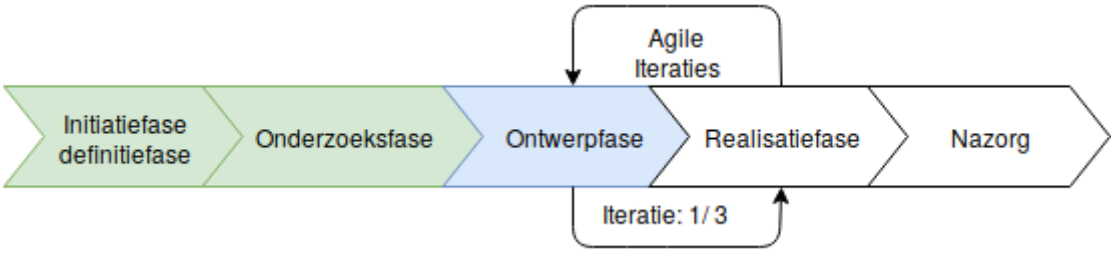
Table 7.1: Resultaten product selectie / aanpak

Zoals te zien in tabel 7.1 bleek uit het onderzoek dat van de drie, Ansible er het best uitkwam. Deze voldeed aan de meeste eisen, had een gratis versie, was open source en had een actieve gemeenschap (community). Het probleem dat bij elk van de softwarepakketten naar voren kwam was een limiet in flexibiliteit als het aankwam op nieuwe netwerken/systemen of flexibele omgevingen.

Secura zal de tool gebruiken in bedrijfsomgevingen waarvan het ontwerp of de omvang niet van tevoren gedefinieerd is (binnen de tools). Met dit in gedachte is ervoor gekozen om zelf een tool te ontwikkelen in Python. Voor elk van de cloud omgevingen is een ‘Software Development Kit’ beschikbaar gesteld door de bedrijven waarmee de API’s simpel aangestuurd kunnen worden en aan alle haalbare eisen voldaan kunnen worden. Daarnaast is zowel de tool als de SDK’s later uitbreidbaar door Secura indien er nieuwe functionaliteit toegevoegd moet worden omdat deze open source zijn (de SDK’s). De taal is gekozen in overleg met de bedrijfsmentor aan de hand van ervaring die de stagiair en de medewerkers van het software ontwikkel team.

7.2. Nessus en Openvas

Tijdens de onderzoeksfase is er tijd besteed aan het onderzoeken naar Nessus/OpenVAS als tools voor het automatiseren van pentests. Het uitzoeken wat deze tools te bieden hebben is gedaan omdat ze populair zijn binnen het securitygebied voor het automatiseren van ‘normale’ pentests op servers en mogelijk gebruikt kunnen worden binnen de cloud. Het onderzoek heeft zicht gericht op de mogelijkheid dat er misschien functionaliteit bestond die (gedeeltelijk) gebruikt kon worden voor het uitvoeren van cloud pentests waardoor mijn opdracht kon werken aan de integratie van de plugin/functionaliteit of kon focussen op een ander onderdeel. De conclusie hiervan is dat er plugins en scripts bestaan die interactie hebben met verschillende cloud omgevingen maar dat deze erg basaal zijn en gelimiteerd zijn in de functionaliteit. Ze zijn te kort door de bocht voor de functionaliteit die Secura wilt implementeren en zijn daarom geen geldige invalshoek (voorlopig). Dit vloeit gedeeltelijk voort uit het feit dat OpenVAS en Nessus bedoeld zijn als test tools voor ‘black box’ pentests waarbij niets over het systeem bekend is. Terwijl wij werken met een Crystal box concept waarbij niet alleen alles weten maak ook toegang hebben tot alle configuraties van de systemen.



8. ITERATIE 1: (PROOF OF CONCEPT)

In het onderzoek is aangetoond dat de cloud instanties aangestuurd kunnen worden met behulp van API's. Met behulp van de bestaande software development kits is het daarom mogelijk om te beginnen met het opbouwen van een tool dat in staat is om automatisch elementen van de cloud omgeving aan te sturen. Deze tool zal als grondlegging van het eindproduct dienen en functioneren als bewijs dat de resultaten van het onderzoek / onderzoeksfase functioneel uitgevoerd kunnen worden.

Dit product kan erg simplistisch uitgevoerd worden om aan de minimale eisen te voldoen, maar met de visie op de volgende producten/iteraties zal het proof of concept proberen om zo veel mogelijk klassen en functies te verwerken die later gebruikt kunnen worden. Hieronder is een lijst gegeven van de verschillende eisen waar dit product aan voldoet. Deze kunnen ook gedeeltelijk voldoen en zullen zichzelf in een latere iteratie verder ontwikkelen.

Voldoet aan:

1. Kan een lijst van systemen en diensten produceren binnen minimaal één cloud omgeving.
2. Kan PaaS configuraties ophalen uit de cloud.

Voldoet gedeeltelijk aan:

6. Kan modulair nieuwe baselines, software en operatie systemen toevoegen.
7. Kan baselines definiëren aan de hand van de softwareversie, OS of Service.
8. Kan de waardes van configuraties / bestanden vergelijken met een baseline.

Om het eindproduct te realiseren zijn de 'Must Have' eisen van belang. Om aan al deze eisen te voldoen in het eindproduct moet het PoC in een vroeg stadium aantonen dat deze haalbaar zijn. Als de 'kerneisen' niet haalbaar zijn kan het project misschien niet uitgevoerd worden naar wens van de opdrachtgever.

8.1. Ontwerp

Het programma zal object georiënteerd geschreven worden om modulariteit te bevorderen. De klassenstructuur die daarom in het ontwerp gebruikt zal worden bij dit project is gebaseerd op de 'objecten' binnen het systeem: Een Cloud serviceprovider biedt diensten aan. Elke cloud serviceprovider en dienst heeft een enkele configuratie (per dienst/cloud serviceprovider, niet gedeeld). Op basis hiervan kan een lijst opgesteld worden van klassen die overeenkomen met de objecten binnen de realiteit:

- CloudProvider
- Instance
- Config

Instance is hierbij de term die AWS en Google Cloud veel gebruiken voor hun PaaS en IaaS diensten. Er wordt hier dus gesproken van een EC2 Instance (IaaS)(waarbij EC2 staat voor "Elastic Computer Cloud"). Dit moet niet verward worden met een instantie van een object of klasse. In de tekst zal vooral de term 'dienst' gebruikt worden om deze verwarring te voorkomen.

De structuur van het ontwerp wordt gekozen op basis van de relaties die zich in de realiteit voordoen. Een dienst (Instance) bevind zich binnen een CloudProvider en is hiervan afhankelijk, zonder CloudProvider kunnen er geen instanties bestaan. Hetzelfde geldt voor een Config object. Zonder CloudProvider of Instance kan de configuratie ervan niet bestaan. Er kon daarom geen andere manier bedacht worden waarop de objecten gedefinieerd worden zonder deze integriteit of simpliciteit weg te gooien. Eén van de opstellingen die bedacht was, was om voor elke soort instantie een eigen klasse te maken met verschillende configuraties voor de cloud provider. Een voorbeeld hiervan is hieronder gegeven:

- EC2Instance
- VMInstance
- S3Instance

Maar deze methode zou wegdoen met de 'cloudprovider' die altijd bovenaan staat en zou zorgen voor heel veel code die niet object georiënteerd was. Daarnaast zorgt deze polymorfie ervoor dat de bovenstaande methodiek nog steeds gedeeltelijk terugkomt als subklasse van 'instance'.

Eis "6. Kan modulair nieuwe baselines, software en operatie systemen toevoegen" is één van de eisen van het systeem met als visie dat het systeem flexibel en modulair is voor aanpassingen. Dit vloeit voort uit de wens om in de toekomst nieuwe Cloud providers en diensten (PaaS/IaaS) aan te bieden binnen de tool. Er is daarom gekozen om gebruik te maken van modulaire opstelling van de klassen. Dit houdt in dat elke 'soort' dienst (e.g. EC2 of S3) een eigen subklasse krijgt van Instance. Een voorbeeld van het model is te zien in Afbeelding 8.1 en Afbeelding 8.2.

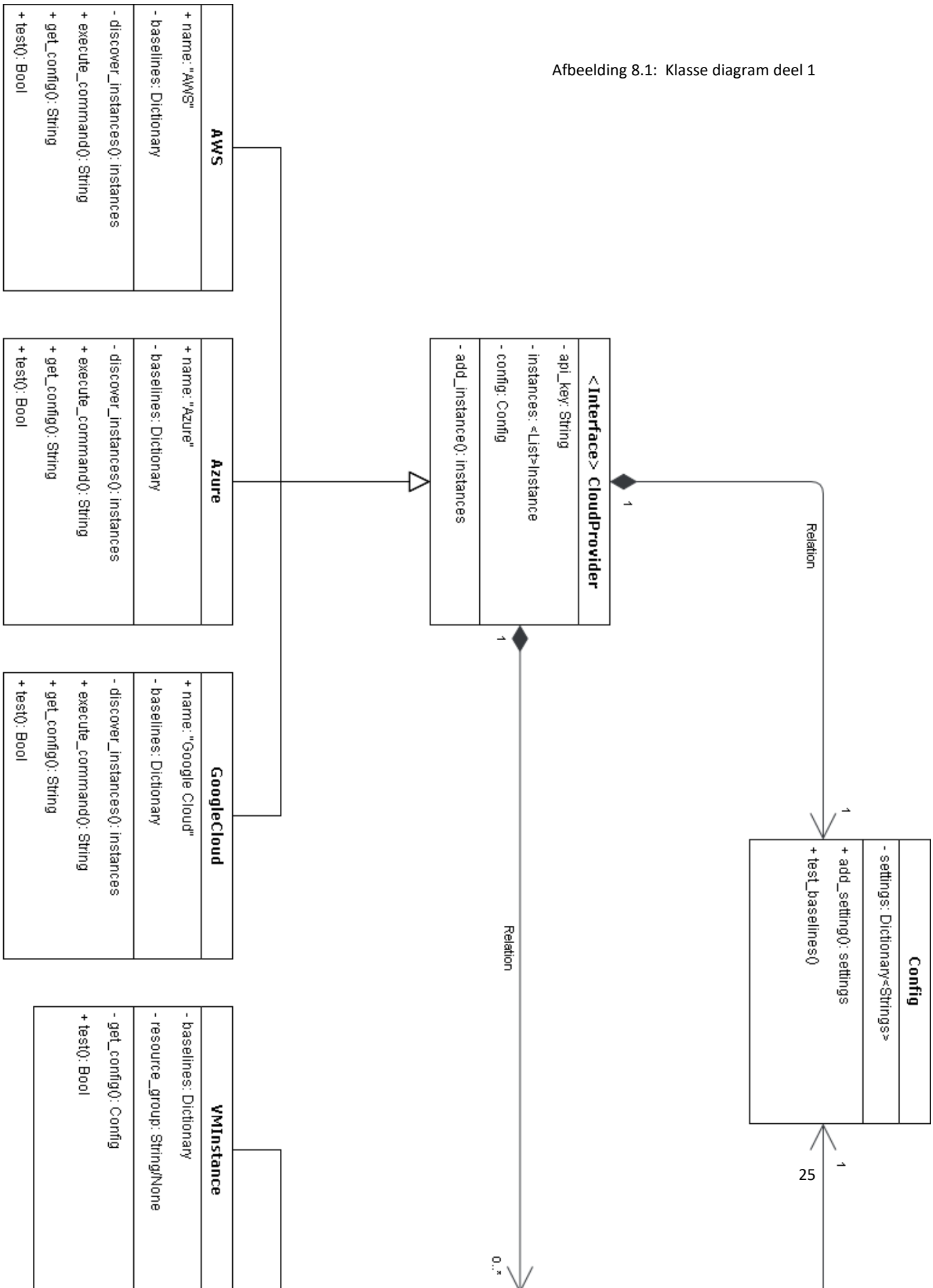
Wat opvalt aan dit ontwerp is dat er 2 'afwijkingen' zijn van een 'standaard' klasse diagram. Ten eerste is er gebruik gemaakt van <Dynamic>. Dit is een 'placeholder' om aan te geven dat de naam van de klasse dynamisch is en aan zal passen aan de hand van het soort klasse dat gemaakt wordt. In het model van het PoC worden er nog voorbeelden toegevoegd zoals "EC2Instance" of "S3Instance". De reden dat <Dynamic> gebruikt wordt binnen het model is omdat het flexibel moet zijn voor de onderhoud baarheid. Op een later moment kunnen er tientallen "<Dynamic> Instance" klassen gedefinieerd worden (AWS beschikt over 173 diensten per 2019-05) om te zorgen dat dit overzichtelijk blijft wordt <Dynamic> gebruikt. Dit zal nog duidelijker worden in volgende iteraties waarbij de complexiteit en formaat van het model groeit (zeker omdat het model – leesbaar – op papier moet passen voor de rapportage).

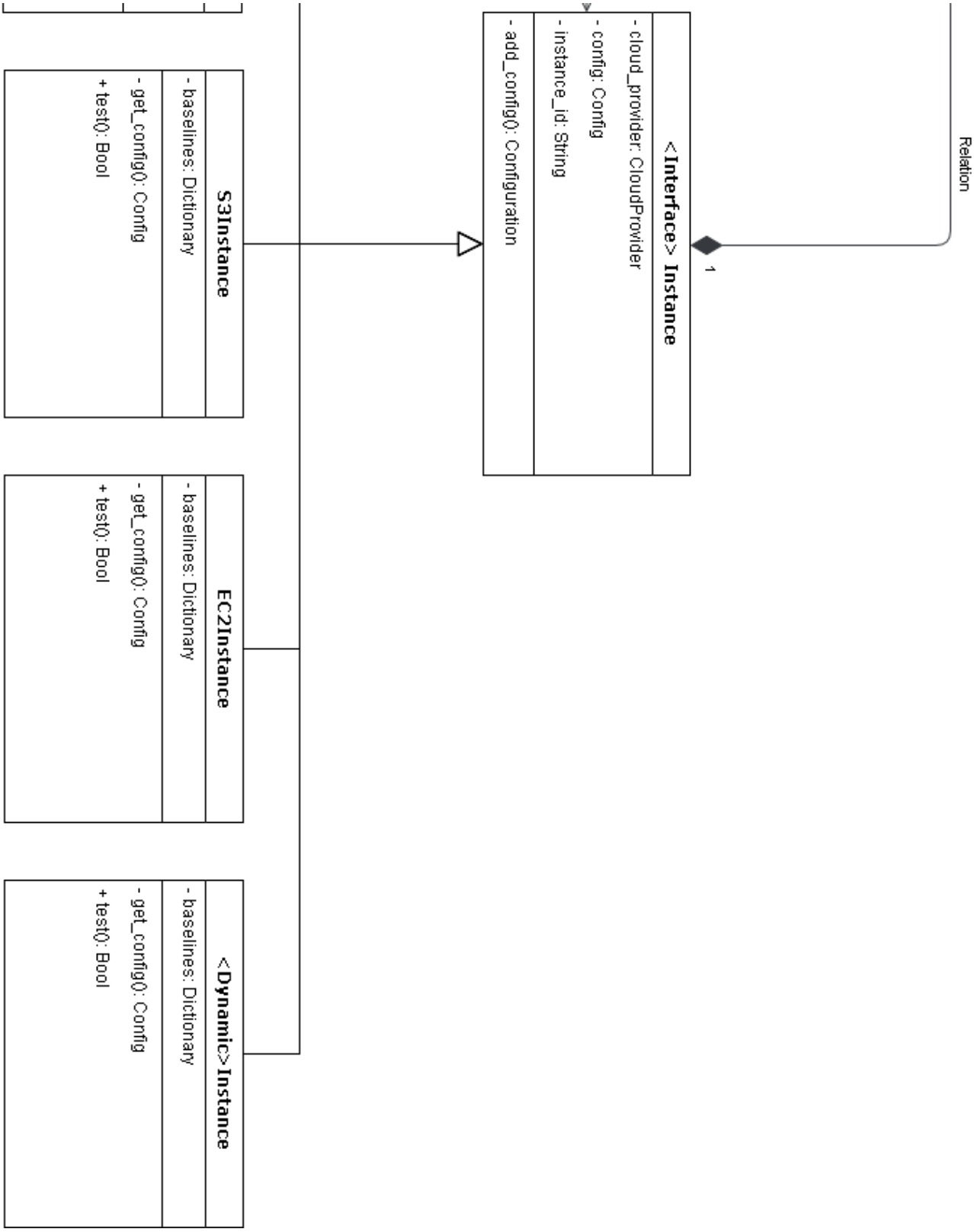
Ten tweede is er sprake van een <Interface> in een python programma. In het geval van dit systeem functioneert dit als een 'interface'-'base class' hybride. Voor dit soort klasse is het zo dat deze nooit aangeroepen mogen worden (zoals een interface) maar wel code bevat die in alle subklasse voorkomt (zoals in een base class). Het is daardoor mogelijk om "add_config" te definiëren in de klasse en te gebruiken in alle subklassen. Terwijl "get_config", die anders is per dienst, opnieuw gedefinieerd moet worden voor alle subklassen omdat deze verschilt per class.

Bij het bouwen van het model is ervoor gekozen om een Config klasse te maken die voor alle onderdelen aangeroepen wordt om instellingen in op te slaan. In het model wordt dit verbonden met de ‘interface’ om aan te geven dat alle subklasse deze verbinding ook bezitten (i.v.m. de bovengenoemde definitie van <Interface>)

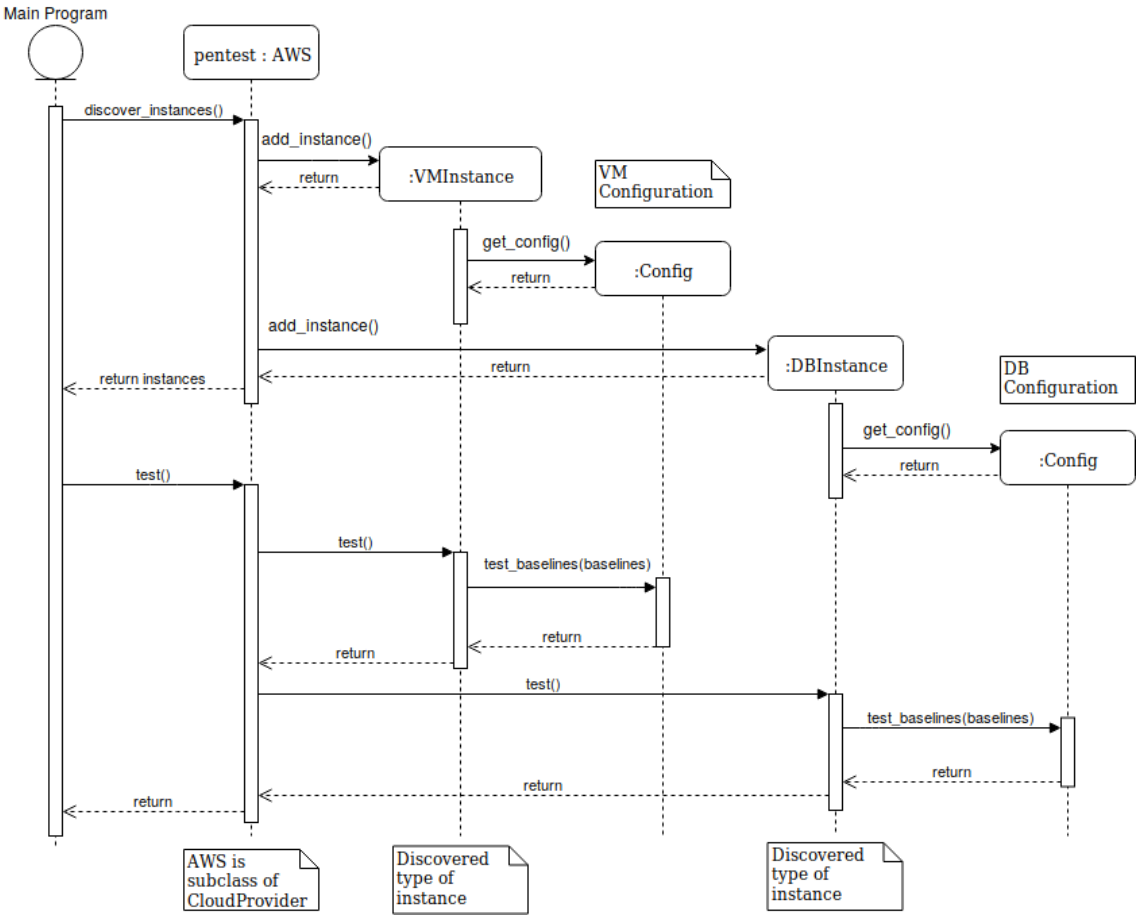
Per klasse wordt een baseline dictionary gedefinieerd dat statisch de ‘baselines’ (richtlijnen) bevat. Er is hiervoor gekozen omdat deze makkelijk aanpasbaar is binnen het bestand, en ze zijn altijd op dezelfde plek beschikbaar, opgesplitst per dienst of cloud serviceprovider. Hoewel dit lastiger onderhoudbaar is, is het specificeren van de baselines per klasse wel schaalbaar en overzichtelijk gestructureerd. De manier waarop het gedefinieerd is stelt een volgende iteratie in staat om deze dictionary te vervangen met een functie die uit een database de informatie ophaalt. Het gebruik van een database centraliseert alle baselines en verbetert de onderhoudbaarheid maar voegt extra complexiteit toe aan een systeem dat alleen bedoeld is om een basis te demonstreren. Er is daarom de afweging gemaakt om voor de eerste iteratie van het programma nog geen database toe te voegen.

Afbeelding 8.1: Klasse diagram deel 1



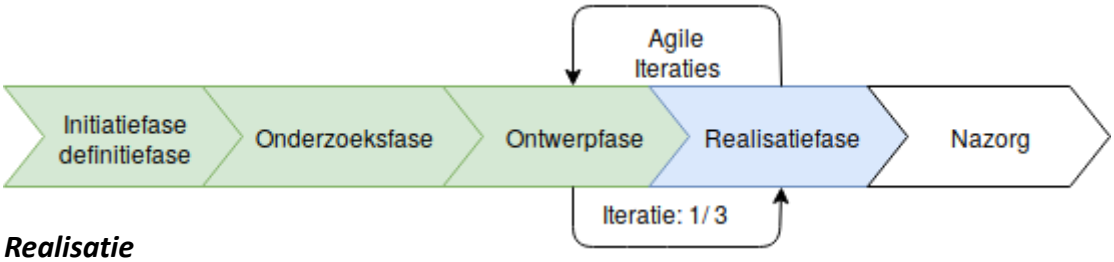


Afbeelding 8.2: Klasse diagram deel 2



Afbeelding 8.3: Sequentie diagram van een security test.

In Afbeelding 8.3 is een sequentie diagram beschreven van een test op een systeem met een VM-dienst en een 'DB' dienst. Deze worden achterhaald door de REST API van bijvoorbeeld AWS te raadplegen. Vervolgens gaan voor elke dienst (Instance) een nieuwe configuratie klasse aangemaakt worden waarbij configuratiewaardes opgevraagd en opgeslagen worden vanuit de API.



8.2. *Realisatie*

Om het ontwerp te realiseren is ervoor gekozen om code te verwerken in een interface. Om dit te realiseren is in afbeelding 8.4 een subklasse te zien genaamd VMInstance. Deze klasse bevat Instance als base klasse en er is in terug te zien dat de init functie van Instance aangeroepen wordt (8.5) Hierdoor kunnen alle baseklasse dezelfde dingen uitvoeren als de 'interface'. Het resultaat is dat subklasse onderdelen aan kunnen roepen zonder ze iedere keer zelf te hoeven definiëren (waardoor de complexiteit van het programma vermindert ten behoeve van eis 6. Kan modulair nieuwe baselines, software en operatie systemen toevoegen).

```
class VMInstance(Instance):
    """Virtual machine subclass of Instance.

    Part of:
      - Microsoft Azure

    Instance is an IaaS style VM; Calls the init function of the parent class (CloudProvider)
    to adhere to OOP design and reduce redundant code.

    Args:
        instance_id (str): name of the VM instance
        resource_group (str): Name of the resource group
        cloud_provider (CloudProvider): Cloud provider instance to run commands

    """
    def __init__(self, instance_id, resource_group, cloud_provider):
        super(VMInstance, self).__init__(instance_id, cloud_provider)
        self.resource_group = resource_group
        if Debug > 0:
            print(Fore.GREEN + "  Found VM instance")
        self.get_config()
        self.baselines = {
            'id': {
                'baseline': 'uid=0\(root\).*?gid=0\(root\)',
            },
            'kernel_release': {
                'baseline': '4.1[45]\..*'
            }
        }
```

Afbeelding 8.4: VM Instance source code.

```
    def __init__(self, instance_id, cloud_provider):
        self.cloud_provider = cloud_provider
        self.config = Config()
        self.instance_id = instance_id

    def test(self):
        self.config.test_baselines(self.baselines)

    def get_config(self):
        raise Exception('Interface should not be initialized directly')
```

Afbeelding 8.5: Instance interface source code (zonder docstring)

```
self.baselines = {
    'id': {
        'baseline': 'uid=0\(root\).*?gid=0\(root\)',
    },
    'kernel_release': {
        'baseline': '4.1[45]\..*'
    }
}
```

Afbeelding 8.6: Baselines voor een EC2 instance.

In afbeelding 8.4 en 8.6 is te zien hoe baselines gedefinieerd zijn in de code. Tijdens het ontwerpen is hier rekening gehouden met hoe dit later uitgebreid wordt. Zo is het overzichtelijk in JSON-formaat opgeslagen en heeft het ruimte om andere ‘velden’ toe te voegen in het datamodel. Zo zou er naast baseline een ‘description’ of ‘verantwoording’ veld toegevoegd kunnen worden vergelijkbaar met en database. Het voordeel hiervan is dat het flexibel is om nieuwe onderdelen te implementeren / experimenteren zonder dat hiervoor grote verandering nodig zijn. Tot slot omdat het model zo ontworpen is kost het weinig moeite om deze functie te vervangen met een database link.

8.3. *Test fase*

Tijdens dit hoofdstuk zullen de verschillende eisen doorgenomen worden waaraan de iteratie moet voldoen, met het doel om te testen of er aan de eisen voldaan wordt door het product. De tests worden uitgevoerd in een testomgeving die opgezet is met 2+ verschillende EC2 / VM-instanties. Deze 2+ instanties waren allemaal Ubuntu 18.04 omdat het proof of concept deze genomen heeft als basis (Arbitraire keuze). Dit is van belang omdat Windows en bijvoorbeeld SUSE andere commando's hebben (zoals apt / yum, Windows heeft geen van beide). Vervolgens wordt er één keer het programma uitgevoerd per test met print statements voor de nodige objecten. Deze tests zijn uitgevoerd in Python3 met een opstelling opgezet volgens het INSTALL.MD bestand in de Github Repository (zie Bijlage D).

1. Kan een lijst van systemen en diensten produceren binnen minimaal één cloud omgeving.

Met behulp van de REST API van zowel Microsoft Azure als AWS is de tool in staat om een lijst op te stellen van instanties per soort service. Volledige objecten per instantie wordt opgehaald waardoor het mogelijk is om niet alleen informatie zoals 'naam', of IP-adressen op te halen maar ook commando's uitvoeren. Dit is van belang omdat dit bijdraagt aan andere eisen.

Stappen:

- Maak een nieuwe EC2 of S3 instantie aan in AWS-controle paneel.
- Creëer 'ec2' of 's3' resource (ec2 = boto3.resource('ec2'))
- Vraag alle instantie objecten op (ec2.instances.all())
- Enumereer de lijst van objecten met een for loop en print de resultaten
- Voer het programma uit

Resultaten:

De twee EC2 s2.micro instanties die aangemaakt waren worden teruggegeven.
De test is succesvol

2. Kan PaaS configuraties ophalen uit de cloud.

Met de objecten die uit Eis 1 gekomen zijn is het mogelijk om configuratie instellingen op te halen van de cloud provider.

Stappen:

- Maak een S3 bucket aan in het AWS-controle paneel
- Creëer 's3' resource (s3 = boto3.resource('s3'))
- Vraag een specifiek onderdeel aan configuratie op aan de hand van de instance ID (van eis 1) zoals encryptie. (s3.get_bucket_encryption(Bucket=instance_id))
- Print het object
- Voer het programma uit

Resultaten:

De S3 bucket die aangemaakt is geeft instellingen terug (zie Afbeelding 8.7)
De test is succesvol

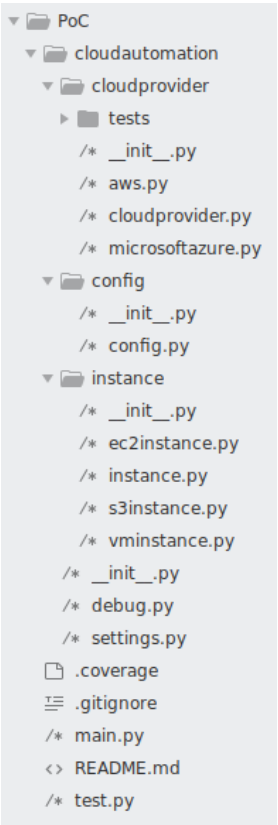
```
{
    'ServerSideEncryptionConfiguration': {
        'Rules': [
            {
                'ApplyServerSideEncryptionByDefault': {
                    'SSEAlgorithm': 'AES256'|'aws:kms',
                    'KMSMasterKeyID': 'string'
                }
            },
        ]
    }
}
```

Afbeelding 8.7 - Resultaten van get_bucket_encryption in JSON-formaat

6. Kan modulair nieuwe baselines, software en operatie systemen toevoegen.

Het modulair zijn van de software wordt mogelijk gemaakt door het object georiënteerd model dat toegepast wordt. Dit is verder verbeterd door alles in een Python package te ontwerpen. Hierdoor zijn alle soorten subklassen georganiseerd in een eigen folder en is het makkelijk om nieuwe onderdelen toe te voegen. Met een korte uitleg zal Secura in staat zijn om gemakkelijk nieuwe functionaliteit aan het programma toe te voegen. De package structuur hiervan is terug te zien in Afbeelding 8.8. (z.o.z.) De 'modulariteit' wordt 'gemeten' in het aantal acties en aanpassingen dat gemaakt moet worden om nieuwe onderdelen toe te voegen. De complexiteit om nieuwe onderdelen toe te voegen is nog hoog en is daarom als 'gedeeltelijk compleet' gemarkeerd. Stappen is hier lastig te meten, maar er moeten meerdere klassen, functies en objecten aangepast en aangemaakt worden om verschillende onderdelen toe te voegen. Hoe 'hoger' het soort object is in het klasse diagram (meer subklasse), hoe lastiger. Dus voor instanties is het makkelijker dan een cloud serviceprovider.

De stappen om nieuwe onderdelen toe te voegen zijn groot, kosten veel veranderingen aan veel verschillende bestanden. De hoeveelheid code die herschreven moet worden is groot, maar wel modulair per definitie. Dus de test is geslaagd, maar hij is niet gemarkeerd als compleet omdat deze significant verbeterd moet worden voordat deze in 'productie' genomen zou kunnen worden.



Afbeelding 8.8: Package structuur PoC

7. Kan baselines definiëren aan de hand van softwareversie, OS of Service

Het toevoegen van nieuwe baselines is geïmplementeerd maar het systeem is nog niet ver genoeg om goed rekening te houden met software of operatie systemen. Met dit in gedachte is deze eis gemarkeerd als 'gedeeltelijk' afgerond. De OS en service kunnen wel als onderdelen van de baseline dictionary toegevoegd worden binnen het programma. Op deze manier is het mogelijk om met een aantal kleine aanpassingen, alsnog de functionaliteit volledig te implementeren (zie 8.2 Realisatie, laatste paragraaf)). Dit is niet gedaan in verband met prioriteit op andere onderdelen.

Stappen om een nieuwe baseline toe te voegen:

- Voeg een nieuwe entry toe aan de baseline dict.
- Voer het programma uit

Resultaten:

De baseline wordt gecontroleerd en een succes/faal bericht wordt getoond.
De test is succesvol

8. Kan de waardes van configuraties / bestanden vergelijken met een baseline.

Het programma is in staat om configuraties te vergelijken met een baseline, maar omdat het bestands onderdeel van deze eis nog niet geïmplementeerd is, is deze baseline as gedeeltelijk voltooid gemarkeerd. Deze beslissing volgt uit de mogelijkheid voor de tool om de configuratie van bijvoorbeeld hardware encryptie op een S3 bucket uit te lezen en te vergelijken met een baseline (Regex string).

Door een PaaS configuratie op te halen zoals in de test van eis 2, en een baseline te definiëren in de S3 bucket zoals in de test van eis 7 kan eis 8 getest worden. Draai het programma na een baseline te hebben gemaakt voor de S3 bucket SSEAlgorithm (zie afbeelding 8.7)

8.4. Demo

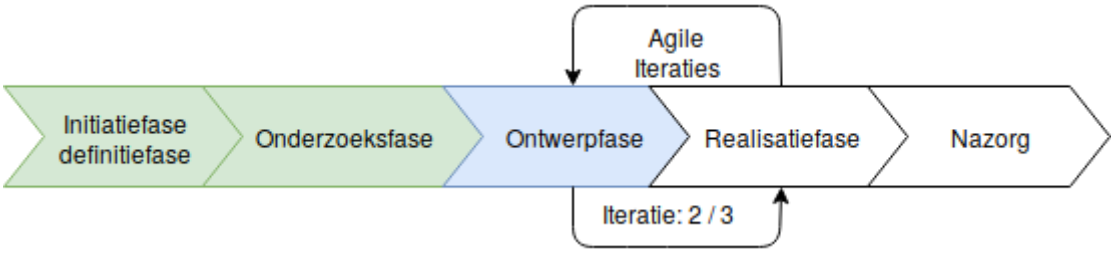
Om deze fase van het project af te ronden is een Demo gehouden met de opdrachtgever op 22 Maart (korte vertraging in verband met een drukke planning). Tijdens deze demo is het er een presentatie gegeven van de functionaliteit van het proof of concept zoals het in GitHub beschikbaar was[8.1]. Tijdens deze demo is behandeld wat de huidige functionaliteit van het product was en hoe het kon voldoen aan de uiteindelijke eisen van het product. De product owner heeft hierbij gemeld dat deze tool uiteindelijk via cron jobs zou moeten kunnen draaien. Dit is een eis die geclassificeerd is als 'Should Have' omdat dit betrekking heeft op het gebruik van het product maar met een lage prioriteit. Deze eis is toegevoegd als nummer 11.

De opdrachtgever heeft zijn akkoord gegeven op het Proof of Concept product op 22 Maart, Locatie Amsterdam.

8.5. Beheersing

Voor beheersing van deze sprint is er een kleine vertraging opgereden in de tijd. De demo is naar achteren geschoven in verband met een drukke planning. Voor beheersing van de kwaliteit van het product is het Proof of concept ook opgestuurd voor een Code Review. Het resultaat van de code review was dat de code er goed uit zag maar dat ik moest opletten met het gebruiken van "magie" zoals: "image_name=result[0][2]['instance']['name'].split('\\\\',1)[0].split('.',1)" waarbij het onduidelijk of onoverzichtelijk is wat het 'precies' verandert of wat het originele resultaat was. (Geen echt voorbeeld)

Aan de hand van het proof of concept en de voldane eisen is te concluderen dat het gebruiken van Python samen met REST API's in staat is om aan de benodigde eisen te voldoen. Dit is gebaseerd op het onderzoek, de tests en code tot nu toe. Dit laat zien dat dat de functies geïmplementeerd kunnen worden en bestaan, zelfs als deze zich niet allemaal in de code bevinden.



9. ITERATIE 2: (PROTOTYPE)

Dit product zal als voortgang functioneren op het Proof of Concept. Het doel is om in dit product (niet iteratie) de kern functionaliteit te verwerken dat het eindproduct nodig heeft en zou zonder veel extra werk door Secura in gebruik genomen moeten kunnen worden als Alpha product indien gewenst. Deze kan dan door pentesters gebruikt worden om te assisteren bij het werk terwijl zij baselines toevoegen om het systeem verder te ontwikkelen (Zie Aanbevelingen).

Er was tijd besteed aan het bouwen van een ontwerp van het Prototype voordat er een gesprek met de begeleider plaatsgevonden had in verband met de drukte. Deze is (Pre-gesprek) genoemd omdat er nog geen gesprek plaatsgevonden had. Vervolgens wordt het 'post-gesprek' ontwerp besproken dat gemaakt is na het gesprek.

Voldoet aan:
3. Kan netwerk configuratie instellingen ophalen uit de cloud.
5. Kan de software detecteren die draait op een IaaS (Apache of Nginx).
6. Kan modulair nieuwe baselines, software en operatie systemen toevoegen.
7. Kan baselines definiëren aan de hand van OS, provider of dienst.
8. Kan de waardes van configuraties / bestanden vergelijken met een baseline.
9. Kan bewijsvoering per baseline genereren (kort).
11. Kan een test via cronjobs uitvoeren.

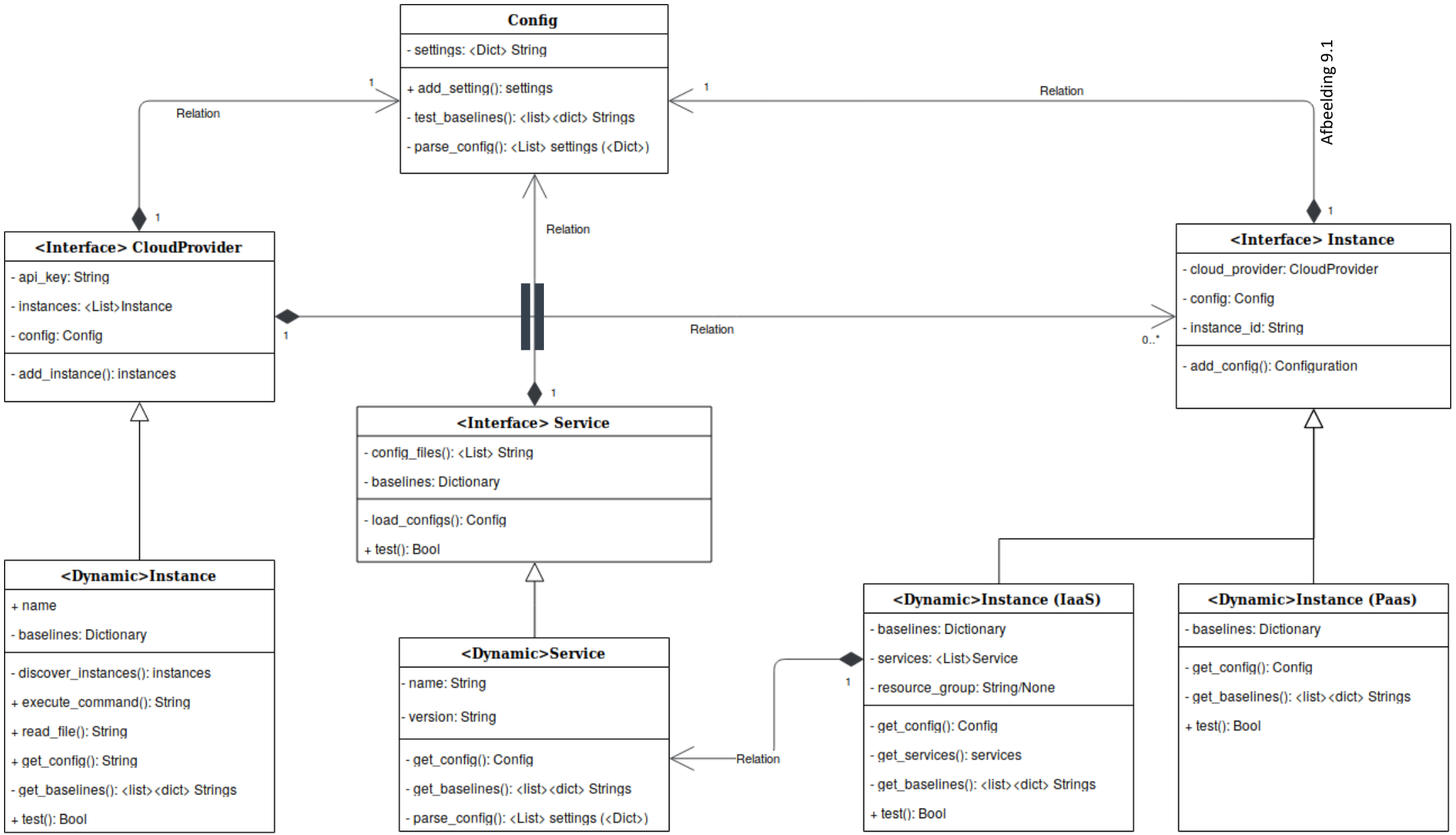
Voldoet gedeeltelijk aan:
13. Kan PaaS diensten controleren op configuratie fouten met baselines.

Niet haalbaar:
10. Kan de output van baselines controles naar de reporting tool sturen met LaTeX. *(Zie 9.6 Beheersing)*

9.1. Ontwerp (Pre-gesprek)

Om nieuwe functionaliteit toe te voegen is er een ontwerp gemaakt dat gebruik maakt van het Proof of Concept als basis. Deze is ook aangepast om de 'drukte' en formaat van het model te verminderen. De voorbeelden voor de verschillende subklasse (AWS, Azure, EC2Instance, S3Instance, etc.) zijn verwijderd waardoor alleen het "Dynamische" object als "placeholder" overblijft. Deze placeholder bestaat omdat er veel verschillende subklasse gemaakt zullen worden nadat deze stage afgerond is. Hierdoor wordt het niet onnodig complex naarmate het model groeit. Ten opzichte van het oude model is een 'service' object toegevoegd omdat elke IaaS instantie meerdere diensten kan bevatten (MySQL, IIS, Apache, Nginx, enz.). Om deze allemaal modulair toe te kunnen voegen is een Service "interface" (Zie H8.1) en dynamic service object toegevoegd (Zie afbeelding 9.1). Voor elke service zal een nieuwe subklasse van 'service' aangemaakt worden als gevolg van dit dynamische object. Deze keuze is gemaakt omdat het object de realiteit representeert. Dit geldt ook voor de relaties die het vormt. Zonder instantie zal een 'service' niet kunnen bestaan.

Het volgende ontwerp is gebouwd vóór het gesprek met de begeleider over het product. Deze bevind zich op de volgende pagina. Het gesprek met de begeleider is later gehouden in verband met de volle planning.



Afbeelding 9.1

9.2. *Ontwerp (Post-gesprek)*

Tijdens het gesprek met de begeleider (Van Secura) kwamen een aantal nieuwe onderdelen naar voren die de gebruikersvriendelijkheid van het project konden verbeteren. In ons gesprek bespraken wij hoe gebruikers het beste nieuwe baselines konden definiëren en welke mogelijkheden Secura de meeste flexibiliteit gaven. Hierbij is een vergelijking gemaakt tussen 'hardcoded', 'JSON-files', 'sqlite database' en een 'SQL-database' voor de opslag van baselines. Van deze mogelijkheden is ervoor gekozen om een SQL-database te gebruiken omdat de opslag ervan goede integratie heeft met bijvoorbeeld webapps. Door de indexering (B-TREE algoritme) van de data is het snel te doorzoeken, zelfs met veel resultaten en hoewel een sqlite database ook de mogelijkheid geeft om snel te doorzoeken zou de SQL-database de keuze geven om een flexibelere webapplicatie eromheen te bouwen die van buitenaf benaderbaar zijn (Zie Aanbevelingen). Dat houdt in dat er maar één versie van de baselines opgeslagen is in plaats van dat elke gebruiker hun versie van de database constant moet bijwerken. De gebruikers zouden daarom op een gestructureerde manier baselines kunnen definiëren vanuit een centrale locatie (zoals een webapplicatie) die daarna niet naar andere gebruikers van de tool gestuurd hoeft te worden. Voor hardcoded ontbreekt deze flexibiliteit compleet en voor JSON-files mist de centralisatie en een deel van de performance die binary tree indexing biedt.

Rapportage

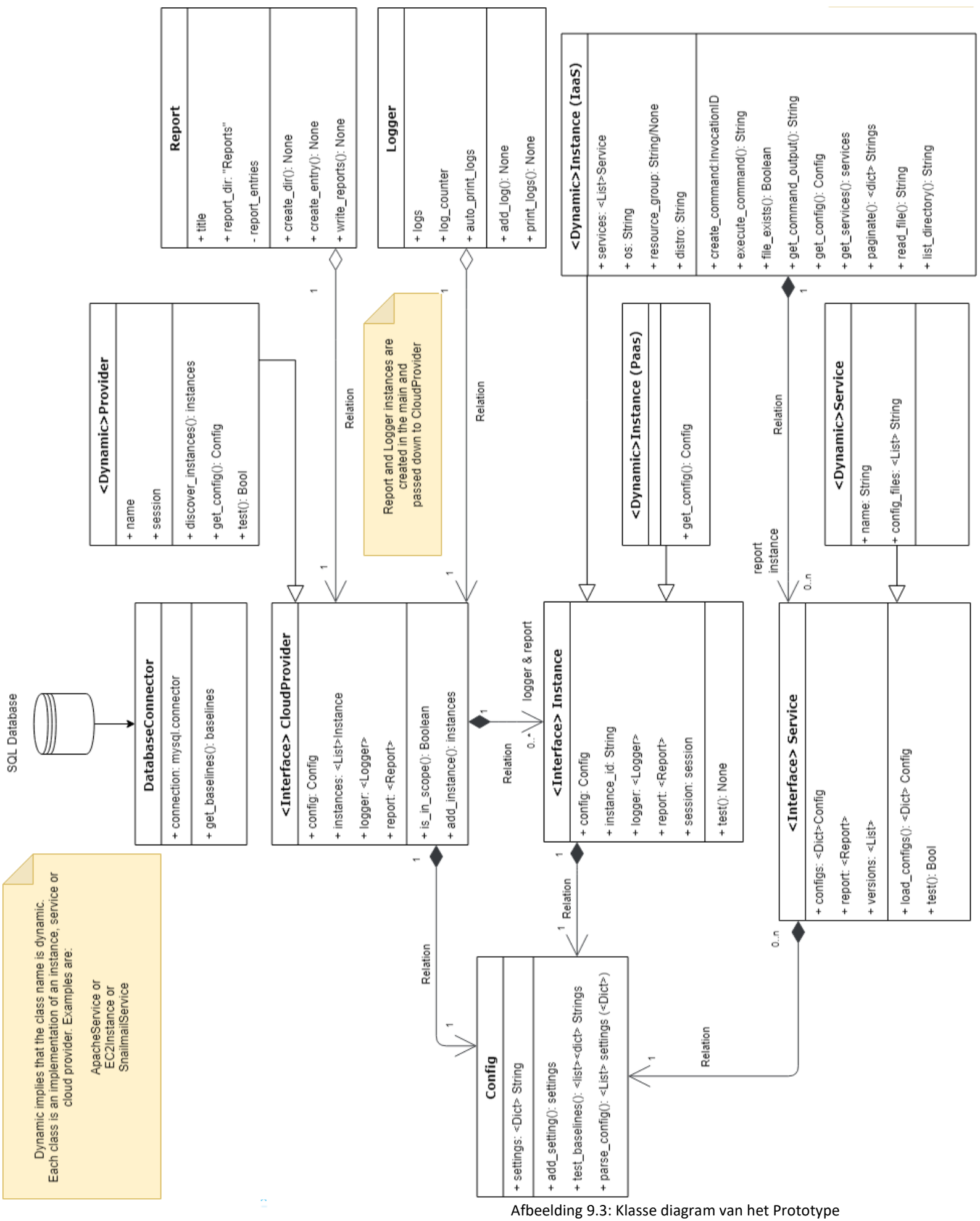
Om rapportage toe te voegen als functionaliteit, is ervoor gekozen om een klasse aan te maken dat in staat is om deze te genereren ongeacht van de 'informatie' dat het toegespeeld wordt. Deze klasse maakt een folder aan waarin JSON-objecten opgeslagen kunnen worden voor rapportage (Zie afbeelding 9.6). Dit maakt het mogelijk om per bedrijf, een sub folder aan te maken waarin de output van de tool opgeslagen kan worden. Om de rol van de tool duidelijk te stellen is ervoor gekozen dat de output van de tool volledig in het JSON zal zijn. Deze universele vorm van opslag maakt het mogelijk voor andere tools om de data in te laden en te formateren naar een gewenste vorm (zoals LaTeX). Een voorbeeld van de rapportage zoals het in de mappenstructuur komt te staan is weergegeven in Afbeelding 9.2.

Afbeelding 9.2: Reports folder structure

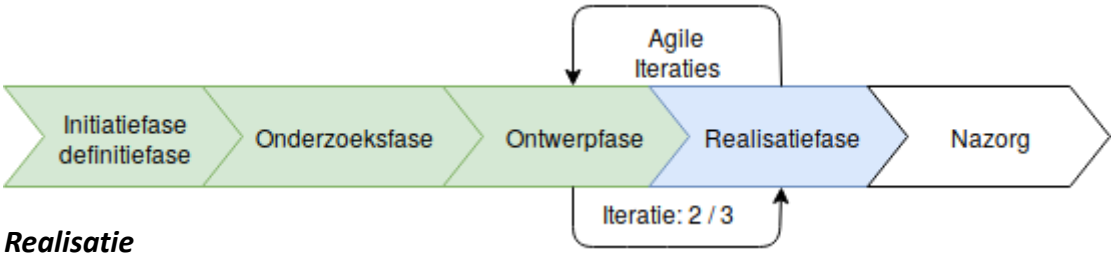
Logger / Logging

Tijdens het ontwerpen van het proof of concept zijn veel informatieve (debug) berichten toegevoegd. Deze herhaalde print statements zijn onpraktisch, zeker als deze met een if/else statement aan of uitgezet moeten worden. De nieuwe klasse lost deze problemen op door als parameters het bericht samen met een 'minimale debug level' te specificeren. Als het debug level te laag is (zoals 0) dan worden er geen berichten geprint door het programma. Dit helpt het bedrijf met debuggen van klanten waar onverwachte resultaten teruggegeven worden en doordat alles gecentraliseerd is maakt de klasse het ook mogelijk om kleine aanpassingen te maken in de logica of output zonder dit voor elke print statement individueel toe te moeten passen.

Het nieuwe model is te vinden op de volgende pagina. Afbeelding 9.3.



Afbeelding 9.3: Klasse diagram van het Prototype



9.3. *Realisatie*

De implementatie in het proof of concept was niet gebruikersvriendelijk omdat hetgeen 'help' commando bevatte en de parameters waren nooit aangegeven (met uitzondering van de code en comments binnen in code). Ook uit de feedback van de opdrachtgever bleek hier nog een probleem te zitten. Om de gebruikersvriendelijkheid te verbeteren is een klasse geïmplementeerd genaamd DocOpt[9.1] dat een makkelijke afhandeling van argumenten mogelijk maakt, controle geeft op gebruikersargumenten en een hulpfunctie aanbiedt. In de onderstaande afbeeldingen (9.4 en 9.5) is het help commando uitgevoerd om inzicht te geven hoe gebruikersvriendelijkheid verbeterd wordt door informatie aan te bieden. De gebruiker kan nu met een duidelijk overzicht zien wat de mogelijkheden zijn die de tool beschikbaar stelt.

```
$ python3 main.py -h
[Secura] Automated Cloud Pentesting Tool

Usage:
  main.py --title=<Company> [--debug=N]
  main.py (-h | --help)
  main.py --version

Options:
  -h --help          Show this screen.
  --version          Show version.
  --debug=N          Debug Level [default: 1].
  --title=Company    Company title [default: "Secura"].
```

Afbeelding 9.4: Programma command line hulp

```
$ python3 main.py --debug=6 --title=test
--debug=N should be integer 0 <= N < 2
```

Afbeelding 9.5: Programma command line, error voor incorrecte debug waarde.

```
"""[Secura] Automated Cloud Pentesting Tool

Usage:
  main.py --title=<Company> [--debug=N]
  main.py (-h | --help)
  main.py --version

Options:
  -h --help                 Show this screen.
  --version                 Show version.
  --debug=N                 Debug Level [0-2] [default: 1].
  --title=Company    Company title [default: "Secura"].

"""
from colorama import init, deinit
from docopt import docopt
```

Afbeelding 9.6: Code om dit te realiseren.

Het prototype maakt met behulp van de Report klasse een map aan dat JSON bestanden bevat (Zie H9.2). Elk JSON-bestand bevat een cluster aan informatie over de omgeving dat vervolgens gebruikt kan worden door een andere tool om het om te zetten naar LaTeX. Een voorbeeld van zo een rapport is te zien in afbeelding 9.7. Dit onderdeel specifiek bevat de routes en subnetten van een AWS-account.

```
},
"sg-0e95c1a3bc3a59ce1": {
    "description": "launch-wizard-1",
    "name": "launch-wizard-1",
    "rules": [
        {
            "from_port": 22,
            "ip_ranges": [
                {
                    "CidrIp": "0.0.0.0/0"
                }
            ],
            "protocol": "tcp",
            "to_port": 22
        }
    ],
    "rules_egress": [
        {
            "from_port": "",
            "ip_ranges": [
                {
                    "CidrIp": "0.0.0.0/0"
                }
            ],
            "protocol": "-1",
            "to_port": ""
        }
    ]
},
"sg-48fa4525": {
    "description": "default VPC security group",
    "name": "default",
```


Afbeelding 9.7: VPC Rapport snippet

9.4. Testfase

Tijdens dit hoofdstuk zullen de verschillende eisen doorgenomen worden waaraan de iteratie moet voldoen, met het doel om te testen of er aan de eisen voldaan wordt door het product. De tests worden uitgevoerd in een testomgeving die opgezet is met 4+ verschillende EC2 instanties en 1+ S3 instantie. Deze instanties waren opgezet met verschillende operatiesystemen waar ondersteuning voor gebouwd was (Ubuntu 18.04, AWS-OS, Windows, SUSE, RHEL, CentOS). Dit is van belang omdat Windows en bijvoorbeeld SUSE andere commando's hebben (zoals apt/yum, Windows heeft geen van beide). Vervolgens wordt er één keer het programma uitgevoerd per test met print statements voor de nodige objecten. Deze tests zijn uitgevoerd in Python3 met een opstelling opgezet volgens INSTALL.MD van de github repository (Zie Bijlage D). Veel onderdelen die door de API opgehaald worden bevatten alle instellingen of extra instellingen die niet relevant zijn. Veel van de code bestaat uit het opschonen van deze data en selecties maken. Eis 10 is niet behandeld in deze fase omdat deze niet haalbaar is, zie H9.6 voor meer informatie.

3. Kan netwerk configuratie instellingen ophalen uit de cloud.

De tool is in staat om verschillende netwerk configuratie instellingen uit de cloud te halen. Hier vallen bijvoorbeeld verschillende virtuele netwerken onder, access control lists of route tabellen.

Stappen:

- Creëer 'ec2' client (client = boto3.client('ec2'))
- Creëer lijst van 'vpc' objecten (vpcs = client.describe_vpcs())
- Loop door "vpcs['Vpcs']" en creëer 'n object voor elk ID (vpc=ec2.Vpc(vpc['VpcId']))
- Print de gegevens van het VPC object uit.
- Haal alle ACL regels op (acls = vpc.network_acls.all())
- Loop door elk ACL object.
- Haal alle Security groepen op (vpc.security_groups.all())
- Loop door elk Security groep object.

Resultaten:

Afbeelding 9.7 geeft een deel van deze objecten weer.
De test is succesvol

5. Kan de software detecteren die draait op een IaaS (Apache of Nginx).

Door te controleren of verschillende bestanden bestaan op het systeem is het mogelijk om te detecteren welke software op een virtuele machine draait. Zo kan het bestaan van het bestand "/etc/apache2/apache2.conf" als bewijs gebruikt worden dat Apache geïnstalleerd is.

Stappen:

- Kijk of een van de configuratiebestanden bestaat (dir /etc/apache2/apache2.conf)
- Geen resultaat of error betekent dat niets bestaat
- Als er iets bestaat haal het versienummer op via command-line (apache2 –V)

Resultaten:

De tool kan detecteren wanneer Apache wel of niet bestaat op het systeem.

- De test is succesvol

6. **Kan modulair nieuwe baselines, software en operatie systemen toevoegen.**
   Met behulp van de nieuwe ontwerpen was het mogelijk om op simpele wijze nieuwe services en onderdelen toe te voegen. Relatief weinig acties moeten ondernomen worden om een nieuw onderdeel toe te voegen en tijdens het zelf testen van dit proces werd dit geclassificeerd als 'simpel' voor veel onderdelen. Zoals een Service (Apache, Nginx) waarbij alleen de naam en bestanden/folders plus een kleine functie geschreven moeten worden voor een complete implementatie. Samen met de nieuwe SQL Database is het ook simpel om nieuwe baselines te definiëren. Er moeten nog steeds kleine aanpassingen gemaakt worden aan de 'core' files bij implementatie van een nieuw onderdeel. De hoeveelheid acties om tot het eindresultaat te komen is verlaagd naar 10 tot 15 afhankelijk van het onderdeel. Hoewel sommige acties nog veel aanpassingen in de code vereisen is de complexiteit dusdanig laag dat deze eis als voltooid wordt gezien. In de volgende iteratie kan deze nog verder verlaagd worden.

7. **Kan baselines definiëren aan de hand van OS, provider of dienst.**
   Tijdens het creëren van nieuwe baselines is het op dit moment mogelijk om te specificeren of deze baseline een AWS, Azure of andere soort provider betreft, welke soort instantie het is (i.e. EC2 of S3) en welk operatie systeem het bij hoort (windows of linux).
   Stappen:
   - Voer een nieuwe insert statement voor de SQL database tabel 'baselines' uit.
     - Deze moet een regex en setting bevatten voor EC2 of S3
   - Voer de test uit met debug level 2 en kijk of de baseline en vergelijking uitgevoerd worden.

   Resultaten:
   De baseline die toegevoegd is wordt uitgevoerd en vergeleken aan het eind van het programma.
   - De test is succesvol

8. **Kan de waardes van configuraties / bestanden vergelijken met een baseline.**
   Met behulp van RegEx vergelijkingen in de SQL-database is het mogelijk om een opgehaalde setting te vergelijken met de baseline uit de database.
   Stappen:
   - Creëer een klasse (zoals de bestaande Apache2) en definieer configuratiefolders of -bestanden.
   - Voer een nieuwe insert statement voor de SQL database tabel 'baselines' uit.
     - Deze moet een regex en setting bevatten voor de relevante klasse.
   - Vergelijk de regex met de waarde van de configuratie bestanden en print de resultaten naar het scherm en rapport.

   Resultaten:
   De tool leest alle gespecificeerde bestanden van Apache uit en haalt configuratie waardes op uit commando's. Deze worden vergeleken met een baseline aan het einde.
   - De test is succesvol

9. **Kan bewijsvoering per baseline genereren (kort).**
   In de SQL-database is het mogelijk om een beschrijving/bewijsvoering te geven voor elke baseline. Dit kan bij de rapporten gebruikt worden om automatisch verantwoording te geven voor de keuze die gemaakt is ten opzichte van de baseline.
   Stappen:
   - Print de uitleg als onderdeel van een rapport

   Resultaten:
   De bewijsvoering / beschrijving van de baseline in de SQL database wordt uitgeprint in het rapport.
   - De test is succesvol

11. **Kan een test via cronjobs uitvoeren.**
    Het programma kan met cronjobs uitgevoerd worden omdat er geen user input nodig is en alle instellingen via bestanden of de commandline meegegeven worden.
    Stappen:
    - Creëer een crontab regel voor het uitvoeren van het programma (1x per dag) “0 14 * * * python3 main.py –debug=2 –company=Secura”

    Resultaten:
    - Het commando wordt om 14:00 uitgevoerd en maakt rapporten aan als ‘bewijs’.
    - De test is succesvol

12. **Kan PaaS diensten controleren op configuratie fouten met baselines.**
    Na het ophalen van alle instanties, is de tool in staat om bestanden uit te lezen en configuraties op te halen uit de API voor elke instantie. Dit stelt de tool in staat om vervolgens de baselines te vergelijken met elk opgehaalde configuratie setting.
    Stappen:
    - Voer een nieuwe insert statement voor de SQL database tabel ‘baselines’ uit.
      - Deze moet een RegEx en setting bevatten voor S3 (of andere PaaS dienst)
    - Maak een client of resource aan voor de dienst
      - (s3_client = self.session.client('s3'))
      - (s3_resource = self.session.resource('s3'))
    - Vraag de bucket encryption op (s3_client.get_bucket_encryption(Bucket=self.instance_id))
    - Sla de setting op (default_encryption_algorithm = server_side_encryption['ServerSideEncryptionConfiguration']['Rules'][0]['ApplyServerSideEncryptionByDefault']['SSEAlgorithm'])
    - En vergelijk de SQL Database RegEx regel met de opgehaalde waarde.

    Resultaten:
    - De output van het script wordt True / False afhangend of er wel of geen encryptie aanwezig is. Met RegEx regel: ((?!None).)*
    - De test is succesvol

9.5. Demo

Op Woensdag 17 April 2019 is een demo gehouden samen met de Opdrachtgever om de voortgang te volgen van het product. Tijdens deze demo is gebruik gemaakt van de Tool met de meest recente versie[9.2]. Samen met de opdrachtgever heeft de stagiair het product doorlopen om de nieuwe functionaliteit te tonen. Hierbij heeft de opdrachtgever aangegeven dat het product goed op weg was en er is een 'akkoord' gegeven op de iteratie. Er is tijdens deze demo aangegeven dat er een code-review gaande was en dat er een akkoord afgesloten was voor het budget.

9.6. Beheersing

In dit hoofdstuk worden de problemen / acties opgeschreven die invloed hadden op de iteratie volgens de GOTIK methode.

Originisatie

Voor de categorie Geld was de 'trial' die beschikbaar was voor Azure verlopen en moest er betaald worden om PaaS of IaaS instanties te kunnen gebruiken. Door bureaucratie en administratie was het pas in eind April mogelijk om een akkoord te sluiten met de CEO voor EUR 100- per maand maximaal. Voor die tijd was het niet mogelijk om te werken aan Azure specifieke onderdelen. Er is daarom een focus gelegd om meer functionaliteit toe te voegen en door te gaan met de gratis instanties die door AWS beschikbaar waren. Er zijn verschillende mailtjes gestuurd naar de begeleider en hoofd van HR; dit duurde langer dan verwacht. Uiteindelijk is dit besproken met de directeur (Dirk Jan) die vervolgend met mij gemaild heeft. Na uitleg vanuit mijn kant waar het geld voor was en hoeveel het zou zijn is er een akkoord samengesteld.

De tool die officiële rapporten genereert en nodig zou moeten zijn voor 'must have' eis 10. "Output van baseline controle in LaTeX naar de rapportage tool.". Was na herhaalde verzoeken nog niet beschikbaar gesteld zonder specifieke reden. Het is daarom niet mogelijk om deze te implementeren of te integreren in het geproduceerde programma.

Kwaliteit

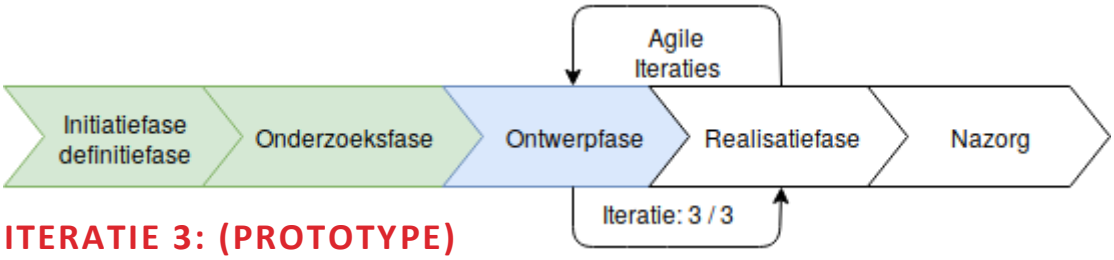
De code review die uitgevoerd is door de development team leader is goed verlopen, hoewel het enige tijd duurde voordat deze afgerond was. De resultaten daarvan zijn kort samengevat in bullet points van 'goed' en 'slecht'.

Goed:

- Gestructureerde code met leesbare layout en design
- Gedocumenteerde code met duidelijke docstrings
- Functionele code zonder linting errors
- Geen problemen met compatibiliteit als het project overgenomen zou worden

Slecht:

- Input van de gebruiker wordt niet altijd gefilterd (Was in eigen code al een TODO bij gezet)
- Aanname dat de data van Amazon altijd correct is (Wat als de API van versie verandert)



10. ITERATIE 3: (PROTOTYPE)

Tijdens de tweede iteratie van het prototype wordt er verder gewerkt aan het product met een focus op afronding en documentatie. Hierbij wordt de feedback van de demo en code review verwerkt en gezorgd dat zowel documentatie van school als van het bedrijf dusdanig op orde is dat het project 'afgerond' is tijdens de nazorgfase. Ook de feedback die ontvangen is van de examinatoren tijdens het TTA en de reviewmomenten worden gebruikt om de documentatie aan te passen. In verband met deze focus en het oog op de testbaarheid van gebouwde of aangepaste functionaliteit zijn er 'extra' eisen toegevoegd die niet officieel vastgesteld zijn, maar wel de wensen van de opdrachtgever, stagiair of code-reviewer weergeven.

Voldoet aan:
6. Kan modulair nieuwe baselines, software en operatie systemen toevoegen. (**Verbeterd**)

Extra:
- E1. Kan commando's uitvoeren via de database en de resultaten weergeven.
- E2. Kan voorkomen dat misbruik gemaakt wordt van command injection / directory traversal.
- E3. Kan testen of een settingswaarde niet bestaat in een configuratie.

10.1. Ontwerp

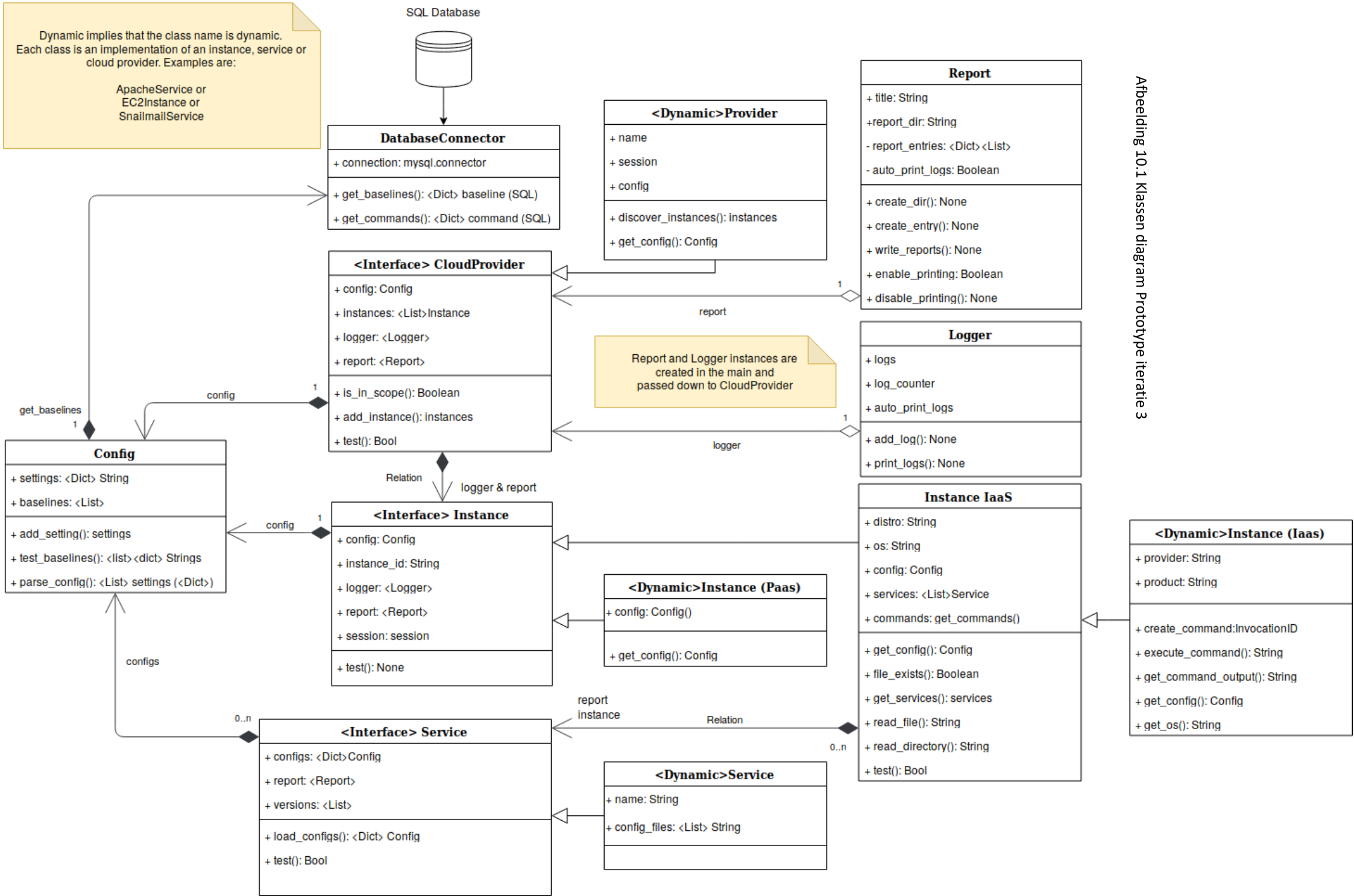
Om de complexiteit van het programma te verminderen en de modulariteit te verhogen is er tijdens het ontwerp gekeken of het mogelijk was om de commando's die uitgevoerd worden ook naar de database te verplaatsen. Hiervoor is 'get_commands()' toegevoegd die vergelijkbaar met 'get_baselines()' informatie ophaalt uit de database in de DatabaseConnector (Zie afbeelding 10.1). De reden dat ervoor gekozen is om commando's naar de database te halen is omdat veel systeembestanden, versie informatie of vergelijkbare onderdelen handmatig erin gezet moet worden. Door dit ook naar de database te verplaatsen wordt er meer vrijheid gegeven aan de gebruikers zonder dat hier de code voor aangepast hoeft te worden. De databasestructuur die hiervoor gekozen is, is gebaseerd op de baselines van de vorige iteratie.

- UUIDv4: uuid
- Varchar(64): setting_name
- Text: command
- Varchar(64): operating_system
- Tinyint(1): split_newline

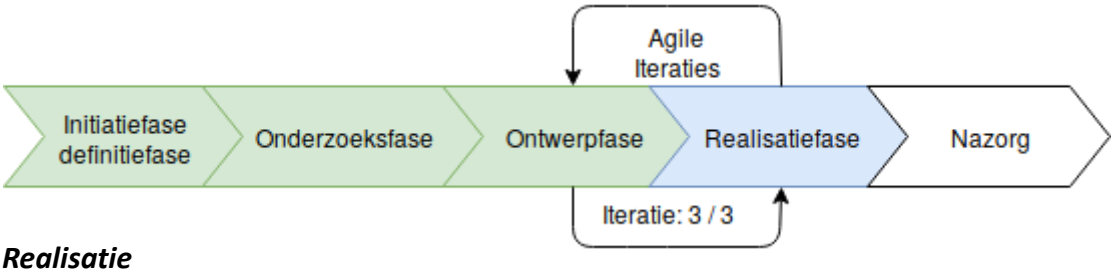
Met dit schema is het mogelijk om nieuwe settings te maken en op te slaan in het systeem.

Door de 'command' waarde uit te voeren op OS de 'operating_system' waarde en op te slaan als de 'setting_name' waarde kan je het commando: "whois" uitvoeren op "windows" en opslaan als "id" terwijl je "id" uitvoert op "linux" en opslaat als "id". Hierdoor kan je twee commando's uitvoeren op verschillende systemen terwijl het in dezelfde locatie opgeslagen wordt. Vervolgens is het mogelijk voor een baseline om te controlereen of de waarde correct is voor linux of windows afhankelijk van welk OS gebruikt wordt.

Om modulariteit te verbeteren is er een nieuwe klasse gemaakt genaamd: "instance_iaas". Deze klasse biedt dezelfde functionaliteit als "instance", het is een interface dat niet geïnitieerd moet worden maar wel code bevat. Deze vorm van polymorfie zorgt ervoor dat code niet hergebruikt wordt waar dat niet nodig is en vermindert de hoeveelheid stappen die ondernomen moeten worden om een nieuwe IaaS instantie klasse aan te maken.



Afbeelding 10.1 Klassen diagram Prototype iteratie 3



10.2. Realisatie

Om de functionaliteit toe te voegen dat SQL-commando's in de database opgeslagen kunnen worden is er een aanpassing gemaakt in de manier waarop de instellingen van een IaaS instantie opgehaald worden. Deze verandering is duidelijker terug te zien in afbeelding 10.3. Deze implementatie ruimt de code erg op ten opzichte van de vorige iteratie (Afbeelding 10.2). Dit is omdat er in een database makkelijk te ordenen commando's toegevoegd kunnen worden met 'eisen' zoals operatie systeem. Terwijl 10.2. (de oude implementatie) vereist dat elk commando zijn eigen codeblock bezit. Dit zorgt voor herhaling, extra if statements en onnodig onderhoud voor alle gebruikers en ontwikkelaars. De nieuwe implementatie verkleint dit allemaal door met een simpele loop door alle commando's in de database te gaan.

```
# Check Who is running the commands
cmd_id = self.cloud_provider.execute_command(self.instance_id, self.os, 'id')
if cmd_id != -1:
    self.config.add_setting(
        {
            'name': 'id',
            'value': cmd_id
        }
    )

# Check the kernel release version
kernel_release = self.cloud_provider.execute_command(self.instance_id, self.os, 'uname -r')
if kernel_release != -1:
    self.config.add_setting(
        {
            'name': 'kernel_release',
            'value': kernel_release
        }
    )
```

Voor Afbeelding 10.2: Code implementatie commando's uitvoeren (individueel)

```
for uuid, command in self.commands.items():
    try:
        cmd_id = self.execute_command(command.get('command'))
        self.config.add_setting(
            {
                'name': command.get('setting'),
                'value': cmd_id
            }
        )
    except Exception as error:
        print('Caught this error: ' + repr(error))
```

Na Afbeelding 10.3: Code implementatie commando's uitvoeren via SQL.

Aan de hand van de code-review die gegeven was tegen het eind van de vorige iteratie, zijn er aanpassingen gemaakt. “Input van de gebruiker wordt niet altijd gefilterd”. Was een van de punten die naar voren kwam als onderdeel van het product dat verbetering vereiste. De reden dat dit een probleem is, is omdat een kwaadaardige of onbekende gebruiker data in kan voeren dat zorgt voor een crash of ongewenste resultaten. Om dit te voorkomen is ervoor gekozen om de data met een RegEx vergelijking te filteren. Deze implementatie is terug te zien in Afbeelding 10.4.

```
def __init__(self, title):
    """Create report if not exists"""

    if not re.match(r'[a-zA-Z0-9\-_]+', title):
        print("ERROR: Invalid company title: '" + title + "'. " +
              "Must be Alphanumeric and can contain -_")
        exit()
    self.title = title  # usually the Company
    self.report_dir = "Reports"
    self.create_dir()
    self.target_dir = self.report_dir + "/" + self.title
    self.report_entries = {}

def create_dir(self):
    self.report_dir = self.report_dir.rstrip('/')
    if not re.match(r'[a-zA-Z0-9\-_]+', self.report_dir):
        print("ERROR: Invalid report directory: '" + self.report_dir + "'. " +
              "Must be Alphanumeric and can contain -_")
        exit()

    if not os.path.exists(self.report_dir):
        os.mkdir(self.report_dir)

    if not os.path.exists(self.report_dir + "/" + self.title):
        os.mkdir(self.report_dir + "/" + self.title)
```

Afbeelding 10.4 Report klasse, create directory en init functie

10.3. Testfase

Deze testfase onderscheidt zichzelf van de vorige iteraties omdat er geen ongemaakte eisen zijn waaraan gewerkt is. In plaats daarvan is er gewerkt aan documentatie, fixen van bugs en opschoning van code. Er zijn ook nieuwe onderdelen toegevoegd waarvan de opdrachtgever van mening was dat ze uitgebreid of verbeterd konden worden, gecombineerd met een aantal ideeën die de stagiair zelf had.

6. Kan modulair nieuwe baselines, software en operatie systemen toevoegen. (Verbeterd)

Door een nieuwe 'base interface' klasse aan te maken is het modulairder geworden, in combinatie met een aantal opschoningen en het updaten van templates. De hoeveelheid acties die ondernomen moet worden is verminderd naar 3-10 waarbij de acties individueel vaak klein zijn. Dit heeft als voordeel dat het nog simpeler is om nieuwe onderdelen toe te voegen en het makkelijker maakt om uit te breiden.

E1. Kan commando's uitvoeren via de database en de resultaten weergeven.

Door commando's toe te voegen en uit te voeren op systemen kan deze baseline getest worden.

Stappen:

- Voeg een command toe aan de 'commands' tabel.
- Voeg een baseline toe aan de 'baselines' met dezelfde 'setting'
- Voer het programma uit en controleer of de baseline een vergelijking maakt met het commando.

Resultaten:

- Het commando wordt uitgevoerd en de baseline vergelijkt de waarde.
- De test is succesvol

E2. Kan voorkomen dat misbruik gemaakt wordt van command injection / directory traversal.

Door de input van de gebruiker te filteren en te zorgen dat alleen bepaalde karakters toegestaan worden kunnen de ze kwetsbaarheden voorkomen worden.

Stappen:

- Start het programma op met company parameters: "../Secura" of "Secura;touch test".
- Test of het programma een error geeft dat de waarde incorrect is.

Resultaten:

- Een error wordt teruggegeven dat ongeldige karakters toegevoegd zijn.
- De test is succesvol

E3. Kan testen of een settingswaarde niet bestaat in een configuratie.

'default values' kunnen aangenomen als waardes niet bestaan in een configuratiebestand. Om dit te testen moet er een error gegeven worden als de RegEx niet voorkomt in een bestand.

Stappen:

- Voeg een baseline toe waarbij de must_contain setting aangevinkt is.
- Draai het programma met een ontbrekende instelling.

Resultaten:

- De baseline faalt omdat de setting niet bestaat en het een waarde verwacht.
- De test is succesvol.

10.4. Demo

Tijdens de SQL meetup op 24 Mei is een presentatie gegeven over de stage. Deze presentatie is afgerond met een demo van het product waarbij gefocust is op de functionaliteit en hoe het eruit zag voordat ik begon, hoe het eruit zag nu ik klaar ben en hoe het er in de toekomst uit kan zien. Feedback die vanuit de opdrachtgever en begeleider komen na dit punt zijn opgeschreven en genoteerd voor de overdracht.

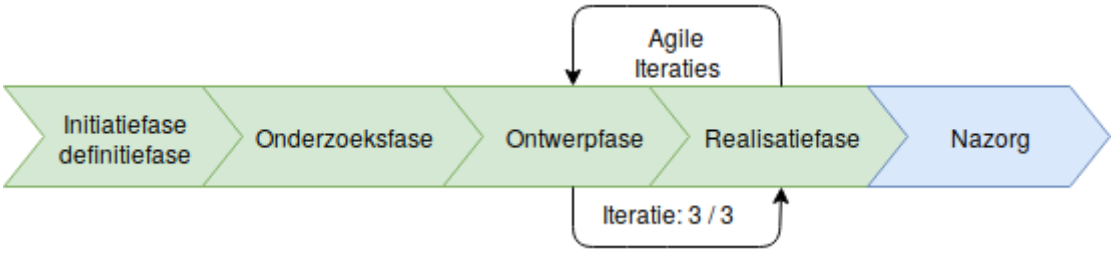
10.5. Beheersing

Voor de beheersing van dit project is er een nadruk gelegd op kwaliteit van documentatie.

Kwaliteit

Om de kwaliteit van het eindproduct en nazorg te beheersen is er tijdens de code review een focus gelegd op de geschreven documentatie voor het project. Hierbij is gekeken naar de documentatie over hoe de ontwikkelomgeving opgezet moet worden, hoe instellingen aangepast moeten worden en hoe nieuwe onderdelen geprogrammeerd kunnen worden. Deze documentatie is terug te vinden in Appendix D.

Tijdens deze periode zijn er meerdere sessies geweest waarbij er feedback geleverd is op de documentatie voor school door de examinatoren. Hierbij zijn grote veranderingen opgetreden in de documentatie. Dit heeft veel extra tijd gekost. Door de ervaring en kennis die opgedaan is op dit gebied zal documentatie in het vervolg minder extra tijd kosten aan het eind van het project.



11. CONCLUSIES EN AANBEVELINGEN

Tijdens de afstudeerstage is de stagiair bezig geweest met het ontwikkelen van een programma dat in staat is om automatische tests uit te voeren in cloud omgevingen met het oog op security. Aan het begin van dit traject heeft de stagiair een onderzoek uitgevoerd naar de mogelijkheden die de populairste cloud diensten aanbieden om informatie te extraheren en systemen aan te sturen. Na het uitzoeken van deze mogelijkheden is de vergaarde informatie gebruikt voor het bouwen van het programma. Het programma en de modulariteit dat het biedt maakt het mogelijk om tests op complete bedrijfsomgevingen uit te voeren zonder de omgeving/lay-out van tevoren te kennen. Dit realiseert de tool door een lijst van alle diensten op te halen en ze stuk voor stuk af te gaan. De tool is in staat om met toegevoegde modules alle PaaS en IaaS instanties te testen op alle onderzochte cloud providers (Google Cloud, Amazon Web Services en Microsoft Azure). Met behulp van RegEx is het programma in staat om de instellingen uit de cloud te vergelijken met gespecificeerde richtlijnen. Doordat de tool modulair opgebouwd is en templates bevat is het simpel om met maximaal 10 stappen een nieuw onderdeel toe te voegen[Bijlage D, DEVELOPMENT.MD].

Als afronding van het stagetraject heeft de stagiair een presentatie gehouden voor alle medewerkers van Secura tijdens de ‘Secura Quarterly meetup’ genaamd “SQL”. Bij deze presentatie is de stage verder uitgelegd en is een demo getoond van het opgeleverde product. Verder heeft er een gesprek met de begeleider en opdrachtgever plaatsgevonden over het product en mogelijke vervolgstappen die genomen kunnen worden om het project verder uit te breiden, en te ontwikkelen naar een volgenden fase. Hierbij is het ‘suggestie document’ gepresenteerd[Bijlage F. Suggestion Report]. In deze periode van afronding is ook de code overgedragen aan het ontwikkelteam met behulp van Gitlab. Doordat er tijdens de code reviews gesproken is over de functionaliteit en werking van het programma hoeft er geen grote uitleg plaats te vinden om van de grond af aan de code en werking uit te leggen. In plaats daarvan wordt deze kort besproken. Ook de documentatie die de code bevat, zowel docstrings als de ‘.md’ bestanden[Bijlage D. Code documentation] zijn gecontroleerd tijdens de code reviews en bevatten naar mening van de Stagiair en Reviewer genoeg informatie om in het vervolg verder te werken aan het project.

Om de tool verder te ontwikkelen, na de afstudeerstage, op het gebied van gebruikersvriendelijkheid en functionaliteit wordt door de stagiair aangeraden om een centrale database te gebruiken met een webapplicatie dat in staat is om nieuwe baselines toe te voegen zonder hier SQL-statements voor op te moeten stellen. Dit maakt het mogelijk om alle medewerkers van het SA-Team dat cloud pentests uitvoert eraan mee te laten werken. Verder zijn er nog meer cloud diensten die toegevoegd kunnen worden om de volledigheid van tests uit te breiden. Deze kunnen niet zomaar gedefinieerd worden zoals de baselines en het development team zal hiervoor de nieuwe onderdelen moeten programmeren. De stagiair raad aan om eerst AWS verder uit te breiden en vervolgens door te gaan met Microsoft Azure. Dit in verband met de relatieve hoeveelheid werk dat erbij gaat kijken om een compleet nieuwe omgeving toe te voegen.

Meer informatie over de aanbevelingen zijn terug te lezen in Appendix F. Suggestie Report.

REFLECTIE

Afstudeerplan

Ten opzichte van het afstudeerplan zijn er veel veranderingen plaatsgevonden. Samen met de aanpassing in de werkmethode is er de keuze gemaakt om van de gekozen producten af te stappen en over te stappen naar een iteratief model. Naar mate de opdracht duidelijker werd, werd het duidelijk dat de opdracht te groot was om productie klaar te maken. Het is daarom voor een vervolgopdracht belangrijk om goed het formaat van de opdracht in te schatten door dat te bespreken met de opdrachtgever.

Plan van Aanpak

Tijdens de eerste fases van het project was er veel onduidelijkheid over de omvang van het project en hoe het precies uitgevoerd zou worden. Dit kwam voort uit een gebrek aan kennis omtrent de cloud diensten en de functionaliteit die zij bezaten. Tijdens de definitie en onderzoeksfase werd veel duidelijk over het soort project dat ontwikkeld zou worden. In plaats van een focus te leggen op de functionaliteit zou de focus liggen op het bouwen van een framework dat in staat zou zijn om alle functionaliteit te ondersteunen dat zo simpel mogelijk aangevuld kon worden. Deze keuze is gemaakt het veel tijd zou kosten om alle baselines op te stellen en te programmeren zonder framework of voorkennis. Gekoppeld met het feit dat er honderdduizenden mogelijke baselines zijn voor duizenden stukken software en PaaS diensten zou het de hele stage opvullen zonder werkelijk iets ‘nuttigs’ te doen.

Tijdens het plan van aanpak was gekozen voor het V-Model en hoewel dit in het verslag ook weerlegd is zal ik nog een korten uitleg geven. Na het gebruiken van het V-Model voor de eerste ‘iteratie’ werd het duidelijk dat het niet flexibel genoeg was om praktisch toegepast te worden op de hele stage. In plaats daarvan zou de methode meerdere malen doorlopen worden. Om dit te veranderen is er gekozen om Agile te gebruiken in verband met de vergelijkbaarheid en de toegevoegde flexibiliteit. Hierdoor kon er een focus gelegd worden op het product met als toegevoegde waarde dat er maar één Agile methode doorlopen werd met drie verschillende iteraties, een start en een einde. Dit versimpelde veel documentatie en gaf een duidelijker overzicht aan de stage.

Bij de opbouw van het plan van aanpak zijn verschillende competenties vastgesteld die tijdens deze stage uitgevoerd zouden worden. Deze zijn weerlegd in Appendix E. Competenties.

Proof of Concept

Toen het proof of concept ontworpen moest worden is er gebruik gemaakt van een klasse diagram om dit goed in kaart te brengen, dit, in combinatie met het vooronderzoek gaf een duidelijk overzicht. Maar tijdens de testfase werd het duidelijk dat de gekozen verwoording voor de eisen lastig opgesteld waren om goed functioneel te testen op het voldoen van de eis. Om dit in een volgend project beter aan te pakken zou ik ervoor kiezen om de eisen SMART(er) op te stellen.

Geldakkoord

Tijdens de opdracht was mijn taak om een geldakkoord af te sluiten voor het gebruik van virtuele machines in cloud omgevingen. Om dit aan te pakken verwachtte ik dat dit via de begeleider met de juiste persoon besproken zou worden. Ik heb daarom aan het begin van de stage aangegeven aan de begeleider dat ik dit graag in werking zou willen stellen, maar zonder haast omdat er gratis krediet beschikbaar was en veel van de eerste weken bestonden uit onderzoek. Vervolgens zijn er meer reminders gestuurd voor deze actie maar het bleek uit een absentie van reactie dat dit niet zou werken in verband met de drukke planning. Op suggestie van de Opdrachtgever heb ik vervolgens contact opgenomen met de team leader van Human Resources, die in bezit was van een bedrijfscreditcard. Na een aantal e-mails tussen elkaar was het mogelijk voor haar om mijn verzoek door te sturen naar de directeur van het bedrijf. De directeur heeft vervolgens een aantal vragen gesteld over waar het voor nodig was, wat de kosten zouden zijn en hoe ik dit aan wou pakken (budget). Vervolgens is er een Akkoord geweest op een budget van EUR 100,- per maand.

Ondanks dat het gelukt is om alles af te sluiten heb ik het gevoel dat ik te rustig geweest ben met het navragen en opvolgen. Uit respect voor de drukte van de medewerkers heb ik meerdere dagen gewacht tussen mails of verzoeken voordat er verzoeken gemaakt werden op vervolging. En door de optie om zonder budget door te werken was er weinig druk om hier snel achteraan te gaan. Waarschijnlijk zou het voor mij ook een goede keuze zijn om aan meerdere mensen dezelfde vraag te stellen tegelijkertijd om sneller de juiste persoon ervoor te pakken te krijgen.

Prototype

In de fase waarbij het Prototype ontwikkeld werd zijn er twee momenten geweest waarbij gewerkt is aan het ontwerp. Hoewel het goed was om het ontwerp opnieuw op te zetten aan de hand van de verkregen feedback had een deel daarvan voorkomen kunnen worden door zelfinitiatief te tonen in de implementatie van een aantal onderdelen die voor de hand liggend waren. Zo was het gebruik van een SQL-database mij al eerder binnengeschoten als optionele implementatie maar zonder prioriteit. Aan de hand van de gesprekken met de begeleider kwam naar voren dat de prioriteit hiervoor hoger lag.

Documentatie

Het schrijven van het afstudeerverslag was een van de grootste leerpunten voor mij. Veel van de onderdelen die in de eerste instantie opgeschreven waren klopte wel, maar waren verspreid over het document. Dit zorgde ervoor dat de onderdelen in verkeerde volgorde behandeld werden. Het overzicht werd al snel gemist en onderdelen ontbraken doordat het werd gezien als ‘voor de hand liggend’ of ‘onbelangrijk’. Na meerdere feedback momenten en het TTA is het hele document significant herschreven. Door het document ook projectmatig in te vullen werd de structuur van het project duidelijker. Door alle evenementen en gebeurtenissen met meer details in te voeren werden de acties die ondernomen waren om het project uit te voeren zichtbaar voor de lezer.

REFERENTIES

[2.1] Secura. “House of Secura” [On-Line]. Available: https://www.secura.com/pathtoimg.php?id=1419 [22-04-2019]
[2.2] Secura. “Organisatie diagram” [Intranet]. [Unavailable] [22-04-2019]
[3.1] CIS. CIS Benchmarks [On-Line]. Available https://www.cisecurity.org/cis-benchmarks/ [13-05-2019
[4.1] N/A. “Agile workflow” [On-Line]. Available: https://i1.wp.com/number8.com/wp-content/uploads/2017/03/Agile-Software-Development.png?fit=607%2C597&ssl=1 [30-04-2019]
[5.1] Appendix C. Haalbaarheidsonderzoek
[7.1] Amazon. (Aug. 2016). “AWS Security Best Practices”. [On-Line]. Available: https://d1.awsstatic.com/whitepapers/Security/AWS_Security_Best_Practices.pdf [11-03-2019]
[7.2] Amazon. “IAM Best Practices”. [On-Line]. Available: https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html [11-03-2019]
[7.3] Amazon. “Boto 3 documentation”. [On-Line]. Available: https://boto3.amazonaws.com/v1/documentation/api/latest/index.html
[7.4] Microsoft. “Azure for Python developers”. [On-Line]. Available: https://docs.microsoft.com/en-us/python/azure/?view=azure-python
[7.5] Kenneth Reitz (2016). “Structuring Your Project”. [On-Line]. Available: https://docs.python-guide.org/writing/structure/ [14-03-2019]
[7.6] Microsoft. “Azure SDK API documentation for Python”. [On-Line]. Available: https://docs.microsoft.com/en-us/python/api/overview/azure/?view=azure-python [13-03-2019]
[8.1] Gitlab commit 06fe419d. [Intranet]. Available: https://gitlab.intern.secura.com/n.loozekoot/cloud-pentest-automation/tree/PoC [02-05-2019]
[9.1] Docopt. [On-Line]. Available: http://docopt.org/ [02-05-2019]
[9.2] Gitlab commit 5cde2aa0. [Intranet]. Available: https://gitlab.intern.secura.com/n.loozekoot/cloud-pentest-automation/tree/Prototype [03-05-2019]