HackDefense

Afstudeerverslag

Password Cracking as a Service

Erik Japing
HackDefense

Februari 2022 – Juni 2022

Final versie 2.0.0

Versiebeheer

Datum	Versie	Aanpassingen
23 maart 2022	0.1.0 - Concept	Opzet document
12 maart 2022	0.2.0 – Concept	Uitwerken inleiding, achtergrond, organisatie
14 maart 2022	0.3.0 – Concept	Uitwerken Probleem/Kans, Juridische aspecten
20 maart 2022	0.4.0 - Concept	Uitwerken Proces – Vergelijkend Onderzoek
21 maart 2022	0.5.0 – Concept	Verder uitwerken Proces
22 maart 2022	0.6.0 – Concept	Uitwerken Proces Ontwikkelen POC
11 mei 2022	0.7.0 – Concept	Uitwerken automatische acties POC
2 juni 2022	0.8.0 – Concept	Uitwerken reflectie
12 juni 2022	1.0.0 – Pre-final	Feedback stagebegeleider verwerkt
14 juni 2022	2.0.0 – Final	Laatste controle, kleine correcties

Voorwoord

Dit document betreft het afstudeerverslag van de opdracht “Password Cracking as a Service”. Tijdens deze opdracht zijn er verschillende onderzoeken uitgevoerd met betrekking tot het opzetten van een infrastructuur voor een wachtwoordkraakdienst, een vergelijking van verschillende kraakprogramma’s en de ontwikkeling van een “proof of concept” waarin het geheel werkend kan worden gezien.

De opdracht is uitgevoerd voor mijn afstuderen van de opleiding Forensische ICT op de Hogeschool Leiden. De opdracht heeft gelopen van februari 2022 tot en met juni 2022.

Ik wil Edward van Veggel bedanken voor zijn rol als afstudeerdocent voor mijn opdracht. Hiernaast wil ik ook de medewerkers van HackDefense zelf erg bedanken voor de gezellige afstudeerplek en de behulpzaamheid. Hierbij wil ik in het bijzonder Sander Meijering en Niels Eriş voor hun rol als bedrijfsbegeleiders en hun inzet om mij te helpen bij het uitvoeren van deze opdracht. Ik kijk ernaar uit om binnenkort als werknemer deel uit kan maken van het HackDefense-team.

Verder wens ik u veel leesplezier toe.

Erik Japing

Den Haag, 13 juni 2022

Inhoudsopgave

1 Inleiding

In dit document wordt de afstudeeropdracht “Password Cracking as a Service” beschreven. Dit document is een samenvatting van de verschillende onderdelen die bij de afstudeeropdracht aan bod zijn gekomen. Op verschillende locaties wordt er in dit document verwezen naar verschillende bijlagen. Deze bijlagen bevatten vaak een uitgebreidere beschrijving van het besproken onderdeel.

Het doel van dit document is om het gehele proces van de afstudeeropdracht te beschrijven en op de bijbehorende werkzaamheden te reflecteren. Hierbij wordt aangetoond hoe er door het uitvoeren van de afstudeeropdracht verschillende A-competenties en B-competenties zijn aangetoond.

In hoofdstuk 2 en 3 wordt er informatie gegeven over de context van de afstudeeropdracht en de afstudeerorganisatie waarbij de opdracht wordt uitgevoerd.

In hoofdstuk 4 worden het probleem en de kans van de afstudeeropdracht besproken. Dit hoofdstuk geeft een beschrijving van de verschillende onderdelen die bij de afstudeeropdracht zijn behandeld in de vorm van hoofd- en deelvragen.

Hoofdstuk 5 bevat informatie over de juridische aspecten die mogelijk relevant zijn voor de afstudeeropdracht. Hierbij wordt wetgeving besproken die relevant is voor het uitvoeren van de afstudeeropdracht zelf, maar ook zaken waar de afstudeerorganisatie rekening mee moet houden in het geval dat de dienst “Password Cracking as a Service” wordt aangeboden aan klanten.

Hoofdstuk 6 en 7 bevatten informatie over de aanpak en het proces van de afstudeeropdracht. Hierbij wordt het plan van aanpak samengevat en wordt er een beschrijving gegeven van de daadwerkelijke werkzaamheden van de afstudeeropdracht. Hierbij gaat het voornamelijk om de totstandkoming van het uiteindelijk ontwikkelde product.

In hoofdstuk 8 en 9 worden de resultaten van de afstudeeropdracht beschreven. Hierbij wordt er een samenvatting gegeven van de voltooide onderdelen van de afstudeeropdracht en wordt er een vergelijking gemaakt met het plan van aanpak. De conclusie geeft uiteindelijk concreet antwoord op de hoofd- en deelvragen die eerder in hoofdstuk 4 zijn behandeld.

Tot slot wordt er in hoofdstuk 10 gereflecteerd op het gehele afstudeertraject en de uitgevoerde werkzaamheden. Hierbij worden ook de verschillende A-competenties en B-competenties besproken in relatie to de uitgevoerde afstudeeropdracht.

In de overige hoofdstukken wordt er ruimte gegeven voor een nawoord en een opsomming van de verwijzingen en bijlagen die in dit document aan bod zijn gekomen.

2 Achtergrond

In bijna elke moderne organisatie wordt er gebruikgemaakt van talloze computersystemen. Deze systemen worden gebruikt voor veel verschillende doeleinden en zijn vaak essentieel voor het functioneren van een bedrijf. Vooral bij het verwerken van grote hoeveelheden gegevens zijn computersystemen een essentieel hulpmiddel geworden, waardoor papierwerk in veel gevallen zelfs als ouderwets wordt gezien. Ook op het gebied van communicatie spelen deze computersystemen een grote rol. Door het moderne internet is het mogelijk om live contact te krijgen met mensen over de hele wereld. Dit maakt het bij veel beroepen ook mogelijk om op afstand te werken. De populariteit hiervan is de afgelopen tijd sterk toegenomen (Rijksoverheid, 2021).

Dit digitale tijdperk brengt echter ook een hoop risico's met zich mee. Doordat veel organisaties nu wereldwijd bereikbaar zijn, betekent dit ook dat er wereldwijd kwaadwillenden zijn die een dreiging kunnen vormen. Hierdoor is het soms mogelijk om op afstand systemen plat te leggen, of om vertrouwelijke gegevens buit te maken. Elke gegevensverwerker heeft hierdoor de verantwoordelijkheid om zijn computersystemen goed te beveiligen (Autoriteit persoonsgegevens, sd). Dit wordt ook wel de "verantwoordingsplicht" genoemd. Als een organisatie niet aan deze eisen kan voldoen, bijvoorbeeld door het gebruik van zwakke wachtwoorden, kan de organisatie aansprakelijk worden gehouden wanneer er een beveiligingsincident plaatsvindt. Dit kan hoge boetes tot gevolg hebben. Een voorbeeld hiervan is een incident bij Transavia in september 2019. Hierbij zijn er persoonsgegevens van ongeveer 83.000 personen buitgemaakt, omdat de organisatie geen sterke wachtwoorden gebruikte voor het beveiligen van de toegang tot deze gegevens (Autoriteit persoonsgegevens, 2021).

Wanneer er persoonsgegevens van een organisatie worden gestolen, kan dit soms ook de inloggegevens van deze gebruiker bevatten. Hieronder vallen soms ook de wachtwoorden of een gehashte vorm hiervan. Hashing is een techniek die kan worden gebruikt om de opslag van wachtwoorden beter te beveiligen. Hierbij wordt niet het wachtwoord zelf opgeslagen, maar er wordt een wiskundige berekening (hashfunctie) op het wachtwoord uitgevoerd die niet teruggedraaid kan worden. Enkel de uitkomst van deze berekening wordt opgeslagen. Wanneer hetzelfde wachtwoord opnieuw wordt aangeleverd, kan er worden gekeken of de hashfunctie dezelfde uitkomst geeft. Als dit het geval is, heeft de gebruiker het juiste wachtwoord aangeleverd. Zo kan een wachtwoord gecontroleerd worden zonder het wachtwoord daadwerkelijk op te hoeven slaan. Als er nu toch persoonsgegevens worden buitgemaakt door een kwaadwillend persoon, zijn de originele wachtwoorden dus niet leesbaar.

Het beschermen van wachtwoorden met een hashfunctie biedt echter geen bescherming tegen zwakke wachtwoorden. Hashes van wachtwoorden kunnen worden gekraakt door de hashes van mogelijke wachtwoorden uit te rekenen en vervolgens te kijken of dit overeenkomt met de al bestaande hashwaarde. Dit vraagt veel rekenkracht, omdat er zo veel wachtwoordcombinaties mogelijk zijn.

HackDefense wil een dienst aanbieden waarbij organisaties hun infrastructuur kunnen laten scannen op zwakke wachtwoorden. Het doel is om te kijken of de bestaande wachtwoordhashes van een organisatie te kraken zijn. Als dit het geval is, is er sprake van een zwak wachtwoord en kan er (automatisch) actie worden ondernomen.

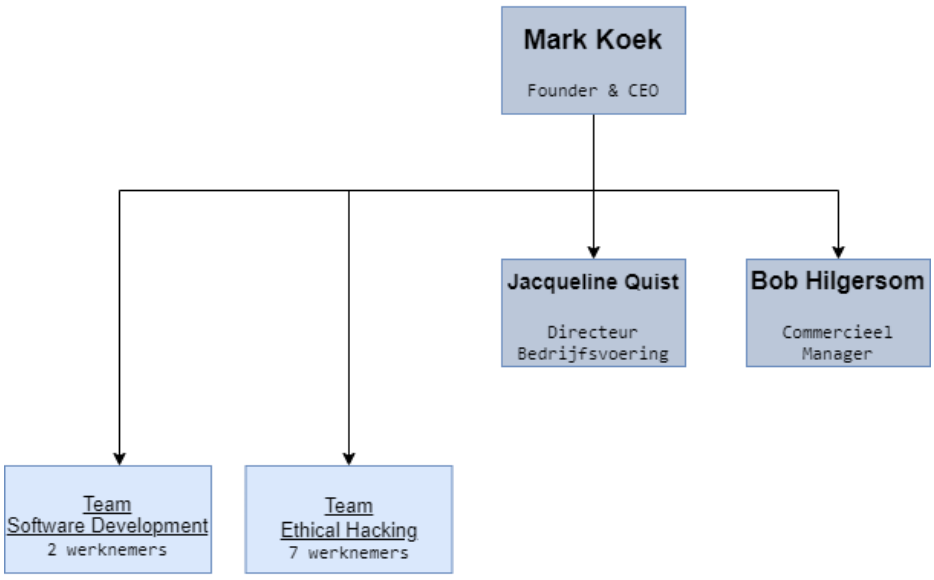
3 Organisatie

Dit hoofdstuk bevat gegevens over de organisatorische aspecten van de afstudeeropdracht. Hieronder vallen bijvoorbeeld contactgegevens van de afstudeerder zelf en van de afstudeerorganisatie.

3.1 HackDefense

HackDefense is een organisatie die zich bezighoudt met IT-beveiliging. De organisatie heeft als doel om de beveiliging van zijn klanten inzichtelijk te maken en advies te geven over de mogelijke verbetering hiervan. HackDefense heeft expertise op verschillende gebieden, waaronder hacking, cybercrime-risico's, software engineering en IT-architectuur. HackDefense geeft niet alleen ondersteuning en advies over technische zaken, maar wil ook dat hun bevindingen begrijpelijk zijn voor het management van een klant. Dit zorgt ervoor dat HackDefense niet alleen helpt met het zoeken van technische oplossingen, maar dat klanten ook worden geholpen met het maken van afwegingen tussen kosten en risico's. Naast het geven van advies en ondersteuning beschikt HackDefense ook over een team van "ethical hackers" voor het uitvoeren van zogenaamde "pentests". Bij dit soort tests wordt er bijvoorbeeld gezocht naar kwetsbaarheden in applicaties, websites of netwerken. Dit soort kwetsbaarheden kunnen worden gebruikt om schade aan te richten of om toegang te krijgen tot bedrijfsgevoelige informatie. Door deze kwetsbaarheden op te laten sporen kan een klant samen met HackDefense een oplossing vinden om kwaadwillenden voor te zijn.

In FIGUUR 1 is een organigram zichtbaar waarin de structuur van HackDefense zichtbaar is. De afstudeeropdracht wordt uitgevoerd onder het team "Ethical Hacking".



Figuur 1, Organigram HackDefense

3.2 Begeleiding en contactgegevens

De afstudeeropdracht wordt uitgevoerd voor de specialisatie **Forensische ICT**. Hierbij is **Jan Hellings** aangewezen als studieloopbaanbegeleider. Binnen deze organisatie is **Sander Meijering** aangewezen als bedrijfsbegeleider voor algemene zaken. Verder is **Niels Eris** ook betrokken als bedrijfsbegeleider voor inhoudelijke zaken. Vanuit de Hogeschool Leiden is **Edward van Veggel** toegewezen als docentbegeleider.

Overige contactgegevens van de genoemde personen zijn hieronder beschreven.

Afstudeerder

Naam: Erik Japing
Adres:

E-mail: S1105925@student.hsleiden.nl
Telefoon:

Studieloopbaanbegeleider

Naam: Jan Hellings

E-mail: hellings.j@hsleiden.nl

Docentbegeleider

Naam: Edward van Veggel

E-mail: veggel.van.e@hsleiden.nl

Afstudeerorganisatie

Naam: HackDefense
Adres: Zijlbaan 28, 2352 BN Leiderdorp

Website: https://hackdefense.nl
E-mail: info@hackdefense.nl
Telefoon: 071-2040101

Bedrijfsbegeleider (algemeen)

Naam: Sander Meijering
Functie: Ethical Hacker/ IT Security Advisor

E-mail: s.meijering@hackdefense.nl

Bedrijfsbegeleider (inhoudelijk)

Naam: Niels Eris
Functie: Ethical Hacker/ IT Security Advisor

E-mail: n.eris@hackdefense.nl

3.3 Stakeholders

Bij de afstudeeropdracht zijn er meerdere belanghebbenden. Deze belanghebbenden worden ook wel “stakeholders” genoemd.

De dienst die tijdens het afstudeertraject wordt ontwikkeld, kan worden afgenomen door verschillende organisaties. Deze organisaties, ook wel “afnemers” genoemd, kunnen gebruik maken van deze dienst om zwakke wachtwoorden binnen hun infrastructuur te detecteren en waar nodig actie te ondernemen. Hiermee kan de afnemer van de dienst zich beter beveiligen tegen mogelijke kwaadwillende gebruikers. De afnemers van deze dienst kunnen hierom als stakeholder worden gezien. HackDefense is voor de start van het afstudeerproject al in contact met mogelijke afnemers van de dienst.

De afstudeerorganisatie, HackDefense kan ook worden gezien als stakeholder van het project. HackDefense heeft baat bij de afstudeeropdracht omdat zij de dienst kunnen aanbieden aan mogelijke klanten. Hiermee kan HackDefense beter bijdragen aan hun doelstelling om kwetsbaarheden bij klanten in kaart te brengen en een completer pakket van diensten aan te bieden. Binnen HackDefense zijn er echter verschillende stakeholders met verschillende belangen. Zo heeft de verkoopafdeling andere belangen dan de pentesters of de directeur van de organisatie. De pentesters zullen de dienst aansturen en hebben dus meer belang bij de functionele aspecten van de dienst. Voor de verkoopafdeling en de directie is er echter ook sprake van een winstoogmerk, aangezien de dienst kan leiden tot nieuwe betalende klanten. De student houdt tijdens de afstudeeropdracht rekening met de verschillende belangen van de medewerkers van HackDefense en neemt dit ook mee in de requirementanalyse.

4 Probleem/Kans

In bijna elke organisatie wordt er gebruikgemaakt van verschillende ICT-systemen. Hierbij worden vaak ook gevoelige gegevens opgeslagen en verwerkt, waardoor het belangrijk is dat deze goed beveiligd worden. Dit houdt in dat deze systemen vaak alleen door bevoegde personen mogen worden gebruikt. Om toegang door onbevoegden te voorkomen, wordt er in de meeste gevallen gebruikgemaakt van gebruikersaccounts die beveiligd zijn met wachtwoorden.

Door gebruikersaccounts te beveiligen met wachtwoorden ontstaat er een grote afhankelijkheid tussen de beveiliging van een systeem en de sterkte van het gekozen wachtwoord. In veel gevallen stellen organisaties eisen aan de complexiteit van wachtwoorden. Denk hierbij aan een minimumlengte, het gebruik van hoofdletters, cijfers of speciale symbolen. Er is echter een groot verschil tussen een complex wachtwoord en een sterk wachtwoord. Dit soort complexiteitseisen kunnen in sommige gevallen zelfs zorgen voor zwakkere wachtwoorden (Carnegie Mellon University, 2020). Ook kunnen wachtwoorden als zwak worden beschouwd als een gebruiker een wachtwoord hergebruikt en eerder is gezien bij een datalek. Nog een andere factor is gebruiksvriendelijkheid. Wanneer een gebruiker last ondervindt van het gebruik van zijn of haar wachtwoord, bijvoorbeeld omdat ze vaak opnieuw moeten inloggen en hieraan veel tijd kwijt kunnen zijn, kan dit ertoe leiden dat een gebruiker geneigd is om gemak te verkiezen boven beveiliging (Agconnect, 2020).

Een organisatie heeft dus maar een beperkte invloed op de sterkte van de wachtwoorden van zijn medewerkers. De organisatie kan eisen stellen aan de complexiteit van wachtwoorden, maar kan moeilijk controleren of deze wachtwoorden ook daadwerkelijk moeilijk te kraken zijn. Indien wachtwoorden op een juiste manier worden opgeslagen (gehasht), is deze opgeslagen versie niet meer te herleiden naar het oorspronkelijke wachtwoord. Deze opgeslagen wachtwoorden zijn dus alleen te achterhalen door ze te kraken, maar de organisatie heeft geen informatie over of deze wachtwoorden wel moeilijk genoeg te kraken zijn.

Een mogelijke oplossing hiervoor is om zelf een test uit te voeren op deze wachtwoorden door ze zelf te proberen te kraken. Als het lukt om sommige wachtwoorden van gebruikers tijdens een test te kraken, toont dit aan dat dit ook mogelijk is voor kwaadwillende personen. Dit soort wachtwoorden kunnen in dit geval als zwak worden beschouwd en vervolgens kan de organisatie samen met de gebruiker van het wachtwoord actie ondernemen.

HackDefense wil een dienst aanbieden waarmee het mogelijk is voor een organisatie om de wachtwoorden van zijn gebruikers op deze manier te testen. Op deze manier kan een organisatie gebruikersaccounts opsporen met wachtwoorden die makkelijk te kraken zijn, zodat deze wachtwoorden kunnen worden aangepast en beveiligingsrisico's kunnen worden voorkomen.

4.1.1 Hoofd-/Deelvragen

Om een overzicht te geven van de verschillende aspecten die tijdens de afstudeeropdracht behandeld worden, zijn de kernpunten onderverdeeld in de onderstaande hoofd- en deelvragen.

- Wat zijn “zwakke” wachtwoorden?
- Op welke wijze kunnen zwakke wachtwoorden worden misbruikt/aangevallen?
- Hoe kan een organisatie zich beschermen tegen zwakke wachtwoorden?
- Wat is “hash cracking”?
  - Hoe vormt “hash cracking” een dreiging voor de beveiliging van organisaties?
  - Wat zijn de beperkingen van deze techniek?
- Welke verschillende methoden zijn er voor “hash cracking”?
  - Wat zijn de verschillen tussen “hash cracking” methodes op het gebied van snelheid en efficiëntie?
- Hoe kan “hash cracking” worden voorkomen om beveiligingsrisico’s tegen te gaan?
- Welke trends zijn er omtrent de bestaande infrastructuur van (potentiële) klanten van HackDefense?

Applicatie

- Hoe kan de applicatie wachtwoordhashes verzamelen uit de bestaande infrastructuur van klanten?
- Hoe kan de applicatie veilig wachtwoordhashes doorsturen naar een (centrale) server?
- Hoe kan een organisatie op een veilige manier worden geïnformeerd worden indien er zwakke wachtwoorden zijn gevonden?
- Hoe kan er automatisch actie worden ondernomen wanneer er een zwak wachtwoord is gevonden?
- Hoe kunnen de resultaten van de “kraaktest” op een gebruiksvriendelijke manier worden gepresenteerd aan de klant?
- Is de applicatie forensisch betrouwbaar?

5 Juridische Aspecten

In dit hoofdstuk worden de verschillende juridische aspecten van het project “Password Cracking as a Service” behandeld. Hieronder vallen juridische onderwerpen die belangrijk zijn voor de dienst zelf, maar ook onderwerpen die belangrijk kunnen zijn voor het doorlopen van het afstudeertraject.

5.1 Wetgeving

5.1.1 Privacy en gegevensbescherming

5.1.1.1 Artikel 8 EVRM

Artikel 8 van het Europees Verdrag voor de Rechten van de Mens geeft “ieder mens het recht op eerbiediging van privé-, familie- en gezinsleven” (Overheid.nl, sd). Dit houdt in dat iedereen het recht op privacy heeft. Dit is ook van toepassing op digitale gegevens (Gerechtshof Arnhem-Leeuwarden, 18).

In Nederland wordt het recht op privacy geregeld in artikel 10 van de Nederlandse grondwet (Artikel 10: Privacy, 2018). De bijbehorende bescherming van persoonsgegevens wordt geregeld in de Algemene Verordening Persoonsgegevens.

5.1.1.2 Algemene Verordening Persoonsgegevens

Tijdens het project “Password Cracking as a Service” wordt er mogelijk met persoonsgegevens gewerkt. Deze persoonsgegevens moeten worden beschermd onder de Algemene Verordening Persoonsgegevens (AVG). Elke gegevensverwerker heeft de verantwoordelijkheid om zijn computersystemen goed te beveiligen (Autoriteit persoonsgegevens, sd). Dit wordt ook wel de “verantwoordingsplicht” genoemd. Als een organisatie niet aan deze eisen kan voldoen kan de organisatie aansprakelijk worden gehouden wanneer er een beveiligingsincident plaatsvindt. Dit kan hoge boetes tot gevolg hebben.

Binnen de AVG wordt er onderscheid gemaakt tussen normale persoonsgegevens en bijzondere persoonsgegevens. Bijzondere persoonsgegevens worden in de AVG gezien als extra gevoelig en moeten daarom extra beschermd worden. De AVG ziet deze persoonsgegevens als bijzondere persoonsgegevens:

- Persoonsgegevens waaruit ras of etnische afkomst blijkt;
- Persoonsgegevens waaruit politieke opvattingen blijken;
- Persoonsgegevens waaruit religieuze of levensbeschouwelijke overtuigingen blijken;
- Persoonsgegevens waaruit het lidmaatschap van een vakvereniging blijkt;
- Gegevens over iemands gezondheid;
- Gegevens over iemands seksueel gedrag of seksuele gerichtheid;
- Genetische gegevens;
- Biometrische gegevens met het oog op de unieke identificatie van een persoon.

Bijzondere persoonsgegevens mogen volgens de AVG alleen worden verwerkt als er hiervoor een wettelijke uitzondering is. Binnen het project “Password Cracking as a Service” is er geen plan met bijzondere persoonsgegevens te werken en er zal tijdens het project aandacht worden besteed om de verwerking van onnodige persoonsgegevens te voorkomen.

Tijdens het project wordt er wel gewerkt met (gehashte) wachtwoorden en mogelijk met gebruikersnamen of pseudoniemen hiervan. Gezien de context van het project moeten deze allebei worden gezien als persoonsgegevens. De reden hiervoor staat verder beschreven in hoofdstuk 5.1. Tijdens het afstudeerproject moet er hierdoor rekening worden gehouden met de AVG. Ook moet het project worden ontwikkeld met aandacht voor de bescherming van persoonsgegevens.

5.1.2 Hacking en Computervredebreuk

Computervredebreuk is het “inbreken” op een computersysteem of netwerk van iemand anders. In veel gevallen wordt computervredebreuk ook wel beschreven als “hacken” of “hacking”. In Nederland is computervredebreuk verboden. Dit wordt geregeld in de wet Computercriminaliteit onder artikel 138ab sr.

Computervredebreuk kan soms als een breed begrip worden gezien. Vaak moet uit jurisprudentie blijken of bepaalde zaken wel of niet als computervredebreuk kunnen worden gezien. Een voorbeeld hiervan is een zaak uit 2005 waarbij valse gegevens werden aangeleverd bij een bank. Dit werd door de rechter niet als computervredebreuk gezien, ook al was er sprake van het aanleveren van vervalste gegevens (Gerechtshof Amsterdam, 2005).

Artikel 138ab geeft vier voorbeelden van computervredebreuk:

- Het doorbreken van een beveiliging;
- Een technische ingreep;
- Valse signalen of een valse sleutel;
- Het aannemen van een valse hoedanigheid.

Het inloggen met een geraden of afgekeken wachtwoord kan worden gezien als het gebruiken van een valse sleutel en kan dus strafbaar zijn (Engelfriet, 2018). In het project “Password Cracking as a Service” wordt er ook gewerkt met het verkrijgen van wachtwoorden van anderen. Hierdoor moeten er duidelijke afspraken worden gemaakt met de afnemers van de dienst, zodat er geen sprake is van wederrechtelijkheid.

In het afstudeerproject wordt er op meerdere momenten mogelijk gewerkt met vertrouwelijke gegevens die gebruikt kunnen worden om schade aan te richten aan de computersystemen van anderen. In artikel 350a sr wordt ook beschreven hoe het strafbaar is om “opzettelijk en wederrechtelijk ter gegevens ter beschikking stelt of verspreidt die zijn bestemd om schade aan te richten in een geautomatiseerd werk” (Maxius, 2016).

Tijdens het afstudeerproject krijgt de student mogelijk kennis over de zwakheid van wachtwoorden van anderen en kan hiermee schade aanrichten aan infrastructuur van anderen. Het vernielen,

beschadigen, onbruikbaar maken of stoornis veroorzaken in de werking van een geautomatiseerd werk is ook strafbaar. Dit staat zo beschreven in artikel 350c sr (Maxius, 2016).

De student moet tijdens het afstudeerproject rekening houden met de verschillende wettelijke eisen die hierbij van toepassing zijn. Het is hierbij belangrijk dat de student zich bewust is van zaken die bij wet verboden zijn en tijdens en het juridisch kader niet overschrijdt. Hierom zal de student tijdens het project kijken naar verschillende manieren om het afstudeerproject aan te pakken en zal alleen kiezen voor methodes die goed aansluiten met de eerdergenoemde wetgeving.

5.2 Zijn wachtwoordhashes persoonsgegevens?

Om de dienst "Password Cracking as a Service" te realiseren, moet er worden gewerkt met wachtwoordhashes. Indien deze wachtwoordhashes op een server van HackDefense worden verwerkt, moet HackDefense worden gezien als een verwerker van deze gegevens. Bij het verwerken van dit type gegevens, is het belangrijk om te weten of hierbij sprake is van persoonsgegevens binnen de definitie van de AVG.

Een persoonsgegeven is "alle informatie is over een geïdentificeerde of identificeerbare natuurlijke persoon". Dit houdt in dat alle informatie die direct over iemand gaat, of indirect tot een persoon te herleiden is, moet worden gezien als een persoonsgegeven. Een wachtwoord zelf moet worden gezien als persoonsgegevens, aangezien wachtwoorden veelal door een gebruiker worden gebruikt om zich te identificeren (Autoriteit Persoonsgegevens, sd). Het maakt hierbij niet uit of een gebruiker direct te identificeren is door een simpel wachtwoord, zoals een naam of geboortedatum, of dat het wachtwoord een willekeurige tekenreeks is die de gebruiker alleen in combinatie met een gebruikersnaam kan identificeren (ICT Recht, 2013).

5.2.1 Anonimiseren en pseudonimiseren

Wanneer informatie geanonimiseerd wordt, is dit niet meer terug te herleiden tot een persoon. Geanonimiseerde data hoeft hierom niet als persoonsgegeven worden gezien en hierdoor is de AVG niet van toepassing op geanonimiseerde data. (Broekhoven, sd)

In de AVG wordt echter wel een definitie gegeven van pseudonimiseren. Bij pseudonimiseren wordt er een aanpassing gedaan aan originele data, zodat deze niet meer terug aan een persoon kan worden gekoppeld zonder dat er aanvullende gegevens worden gebruikt. Er is hier een belangrijk onderscheid tussen het anonimiseren van data, omdat gepseudonimiseerde data nog wel (indirect) te herleiden is tot een persoon. Gepseudonimiseerde data moet dan ook als persoonsgegevens worden gezien onder de AVG. Zowel directe als indirecte herleidbare gegevens moeten dus als persoonsgegevens worden gezien. (PrivazyPlan, sd)

5.2.2 Is een wachtwoordhash anonieme of pseudonieme data?

Voor het project "Password Cracking as a Service" wordt er voornamelijk gewerkt met wachtwoorden in gehashte vorm. Hierdoor is het juridische onderscheid tussen wachtwoorden en gehashte

wachtwoorden een belangrijk detail. Het doel van het hashen van wachtwoorden is om deze op een onherleidbare manier te bewaren. Indien een gehasht wachtwoord op zichzelf wordt opgeslagen, zonder dat er bijvoorbeeld een tabel beschikbaar is waarbij dit wachtwoord te herleiden is aan een bijbehorend emailadres of gebruikersnaam, kan er sprake zijn van anonieme data en is de AVG hier niet van toepassing. Echter wanneer het gehashte wachtwoord, direct of indirect, kan worden herleid tot een persoon, is er sprake van verwerking van persoonsgegevens onder de AVG.

Een andere belangrijke factor bij het verwerken van persoonsgegevens is de context. Binnen het project wordt er een poging gedaan om wachtwoordhashes te kraken. Dit houdt in dat er wordt geprobeerd de hashes terug te draaien naar de oorspronkelijke vorm. Als het lukt om een wachtwoordhash te kraken, is de hash dus teruggedraaid naar zijn oorspronkelijke vorm; het wachtwoord. Dit wachtwoord moet in veel gevallen worden gezien als een persoonsgegeven, omdat het wordt gebruikt om een persoon te identificeren. Hierbij zou er kunnen overwogen om het wachtwoord zelf niet op te slaan, maar slechts de wachtwoordhash te markeren als zwak. Dit geeft echter aan dat het mogelijk is om deze wachtwoordhashes te kraken en terug te draaien naar hun originele vorm. Hierom is er in dit geval dus ook geen sprake van anonieme data, maar pseudonieme data, waardoor de AVG wel van toepassing is.

Kortom, de grens tussen anonimiteit of pseudonimiteit in de context van wachtwoordhashes lijkt nog een relatief grijs gebied. In de context van “Password Cracking as a Service” lijkt het bijna niet mogelijk om echt te spreken van anonimiteit, waardoor wachtwoordhashes wel moeten worden gezien als persoonsgegevens en de AVG hierdoor dus ook van toepassing is.

6 Aanpak

In dit hoofdstuk wordt de algemene aanpak van de afstudeeropdracht besproken. Een uitgebreidere beschrijving van de aanpak is beschikbaar in de bijlage "*Afstudeerplan.pdf*".

De ontwikkeling van "Password Cracking as a Service" is onderverdeeld in een aantal verschillende fasen:

- Aanloop/planning van de opdracht
- Vooronderzoek/literatuuronderzoek
- Uitvoeren van een requirementanalyse en het houden van interviews
- Ontwerpen en opzetten van de infrastructuur
- Het testen en vergelijken van (kraak)software
- Het ontwikkelen van het "proof of concept"
- Het testen van het eindproduct

Door deze indeling in verschillende fases leek ontstond eerst het idee om het product te ontwikkelen volgens de "watervalmethode". Dit is een methode waarbij het ontwikkelproces ook onderverdeeld is in verschillende fases, namelijk analyse, ontwerp, implementatie, testen en onderhoud. Dit lijkt erg op de eerdergenoemde fases, waardoor dit een geschikte methode lijkt.

Bij het uitvoeren van de afstudeeropdracht bleek al snel dat het werken in volledig aparte fases geen goed idee was voor het uitvoeren van het project. Dit komt doordat veel verschillende fases van de opdracht sterk afhankelijk zijn van andere onderdelen. Zo kan het ontwikkelen van het "proof of concept" aantonen dat er verbeteringen in de infrastructuur mogelijk zijn of dat bepaalde beperkingen zijn in kraaksoftware die niet bij de eerste tests zijn opgemerkt. Zo leek het beter om de fases onder te verdelen in verschillende korte iteraties en meer te werken volgens de "agile"-methode. Bij deze methode wordt er gewerkt in korte "sprints". Zo kan er elke sprint een enkele functionaliteit worden ontworpen, ontwikkeld, getest en geïmplementeerd. Hierdoor waren er veel meer momenten om het project bij te sturen en kunnen risico's sneller worden opgevangen.

Sommige fases kunnen wel (bijna) volledig worden afgerond voordat er aan de volgende fase wordt gewerkt. Zo is er tijdens het vooronderzoek/literatuuronderzoek veel informatie verzameld over alle aspecten van het project. Hierbij is er gekozen om te werken volgens de sneeuwbalmethode en het verloop van het onderzoek vast te leggen in een mindmap. Er is hiervoor gekozen omdat er zo met slechts heel weinig informatie kan worden begonnen en het onderzoek langzamerhand steeds meer richtingen krijgt. De resultaten van een zoekopdracht leidt zo tot meer zoekopdrachten. Dit proces blijft zich herhalen en het onderzoek groeit als een "rollende sneeuwbal". Het werken in een mindmap zorgt er hierbij voor dat er geen tunnelvisie ontstaat en er op elk moment kan worden "uitgezoomd" om het overzicht van het onderzoek te behouden. Achteraf bleek het nadeel van deze methode dat het moeilijker kan zijn om een duidelijk eindpunt van het onderzoek te bepalen en de scope beperkt te houden.

Voor het uitvoeren van de requirementanalyse is er gekozen om interviews te doen met verschillende stakeholders. Het doel hiervan is om de wensen vanuit verschillende perspectieven van de organisatie te verzamelen. Zo heeft personeel van sales of van management bijvoorbeeld andere belangen bij het product vergeleken met pentester of potentiële klanten. Deze interviews resulteren in veel verschillende wensen en ideeën. Deze ideeën zijn vervolgens geprioriteerd volgens de MoSCoW-methode. Deze methode zorgt ervoor dat functionaliteiten worden onderverdeeld in verschillende categorieën van prioriteit. Hieronder vallen must-haves, should-haves, could-haves en won't-haves. Deze prioriteiten zijn toegepast op een manier waarop de meest waardevolle ideeën voor de afstudeeropdracht een hogere prioriteit krijgen. Denk hierbij bijvoorbeeld aan functionaliteiten met betrekking op de communicatie binnen de infrastructuur. Ook helpt dit bij het maken van een Minimum Viable Product (MVP). Dit is een product dat nog niet alle functies heeft, maar wel compleet genoeg is om bruikbaar te zijn en om feedback te ontvangen voor de rest van het ontwikkelproces (Agile Scrum Group, sd).

Uitgebreidere informatie over de aanpak is beschreven in de bijlage "*Afstudeerplan.pdf*". Ook is er een verdere beschrijving over het verloop van deze aanpak in het hoofdstuk 10 Reflectie van dit document.

7 Proces

In dit hoofdstuk wordt het proces beschreven van verschillende onderdelen van de afstudeeropdracht. Dit hoofdstuk is ingedeeld in verschillende subhoofdstukken. Elk subhoofdstuk beschrijft hierbij een andere fase van de afstudeeropdracht. Elk subhoofdstuk geeft een korte beschrijving van het gehele proces, waarna er dieper wordt ingegaan op specifieke keuzes of problemen die volgens de student centraal stonden.

7.1 Aanloop en uitvoeren vooronderzoek

Het afstudeertraject begon met het oriënteren op de opdracht en het uitvoeren van een literatuuronderzoek. Het doel van dit onderzoek is om informatie te verzamelen over relevante informatie over de verschillende onderwerpen die bij de opdracht relevant kunnen zijn. Hierbij is er onderzoek uitgevoerd naar verschillende basisonderwerpen, zoals informatiebeveiliging, wachtwoorden en het kraken van hashes. Hiernaast zijn er ook specifieke onderwerpen gekomen vanuit de afstudeerorganisatie zelf, zoals de vraag of wachtwoordhashes als persoonsgegevens zijn en hoe er veilig en anoniem tussen twee apparaten kunnen worden gecommuniceerd. Deze verschillende onderwerpen zijn ook meegenomen in het literatuuronderzoek.

Voor het uitvoeren van het literatuuronderzoek is er gebruik gemaakt van de sneeuwbalmethode. Hierover staat meer beschreven in hoofdstuk 6 Aanpak. Het voordeel van deze methode is dat er met slechts weinig voorkennis kan worden gestart. Er wordt begonnen met een startonderwerp, die dan steeds iteratief leiden tot meer onderwerpen. Zo groeit het onderzoek “als een sneeuwbal”. Het voordeel hiervan is dat een onderzoek zichzelf richting kan geven. De valkuil hierbij is echter dat er tunnelvisie ontstaat omdat er te lang wordt ingegaan op een enkel onderwerp. Om dit te voorkomen is er gekozen om het onderzoek bij te houden in een mindmap. Hierdoor kon de student visueel het verloop van het onderzoek bijhouden en was het makkelijk om onderwerpen terug te herleiden naar eerdere iteraties van het onderzoek. Het gebruiken van deze methode heeft het mogelijk gemaakt om een breed onderzoek uit te voeren, maar ook diepgaand onderzoek toe te passen waar die nodig was.

In het onderzoek is er ook besloten om extra aandacht te besteden aan de vraag of wachtwoordhashes als persoonsgegevens moeten worden beschouwd. Dit is gedaan omdat dit onderwerp ook invloed kan hebben op de selectie van infrastructuur. In het geval dat er met een centrale kraakserver wordt gewerkt, moet het namelijk mogelijk zijn dat de wachtwoordhashes van een klant de organisatie verlaten. Juridisch gezien is het hier nu belangrijk om te weten of er nu sprake is van verwerking van persoonsgegevens. In het onderzoek naar dit onderwerp zijn er een aantal conflicten gevonden. Het is gebleken dat wachtwoorden, in tekstvorm, wél als persoonsgegevens moeten worden beschouwd. Anonieme data worden echter niet als persoonsgegevens gezien. Aangezien een hashfunctie in principe onomkeerbaar is, kunnen wachtwoordhashes in de meeste gevallen worden gezien als anonieme data, waardoor er geen sprake is van persoonsgegevens. In de dienst “Password Cracking as a Service” wordt

er echter gecontroleerd of het wel mogelijk is om de wachtwoordhash terug te draaien naar het originele wachtwoord in tekstvorm. Dit geeft het idee dat wachtwoordhashes in de context van “Password Cracking as a Service” toch wel als persoonsgegevens moeten worden gezien. De student is echter niet opgeleid om volledige uitspraken te doen over dit soort juridische kwesties. Hierom student heeft ervoor gekozen om de complexiteit van dit onderwerp te beschrijven, maar geen definitieve uitspraak of conclusie te geven.

7.2 Houden van interviews

Voor het uitvoeren van de requirementanalyse zijn er eerst interviews gehouden. Hiervoor zijn er verschillende stakeholders van het project geselecteerd. Er is gekozen om interviews te hebben met personen die allemaal verschillende belangen hebben bij het project. Zo hebben verschillende medewerkers binnen de stageorganisatie ook verschillende belangen bij het project. Pentesters hebben bijvoorbeeld hele andere belangen dan bijvoorbeeld management of sales. Dit is verder beschreven in hoofdstuk 3.3 STAKEHOLDERS. Uiteindelijk zijn er interviews gehouden met vier werknemers: Jony Schats, Niels van Gijzen, Bob Hilgersom en Mark Koek. Hiernaast zijn er ook vergaderingen bijgewoond met de projectpartner van de afstudeerorganisatie. Op deze manier is het mogelijk geweest om een breed bereik aan belangen mee te nemen in de requirementanalyse.

Om structuur aan te brengen in de interviews, zijn er voorafgaand de interviews een aantal vragen opgesteld. Het doel hiervan is om de interviews concreet te kunnen houden en ook het risico te verkleinen dat bepaalde onderwerpen worden vergeten. De vragen zijn ook eerst besproken met de bedrijfsbegeleider voor feedback en verificatie. Ook zijn de geïnterviewde personen ruim voor het interview kort op de hoogte gesteld van de verschillende gespreksonderwerpen. Het doel hiervan is om de geïnterviewde personen ook de kans te geven om zich voor te bereiden op het interview.

De totstandkoming van de interviews en het verloop van de interviews zijn beschreven in bijlage “*Requirementsanalyse.pdf*”.

7.3 Requirementanalyse

Om het “proof of concept” te kunnen ontwikkelen moest er eerst een keuze worden gemaakt van functionaliteiten die zouden worden ontwikkeld. Door het houden van de eerdergenoemde interviews zijn er verschillende ideeën en perspectieven verzameld van de verschillende stakeholders van het uiteindelijke product. Deze verschillende ideeën zijn vervolgens ingedeeld op prioriteit. Hiervoor is er gebruik gemaakt van de MoSCoW-methode. Met deze methode zijn de verschillende functionaliteiten onderverdeeld in “must-haves”, “should-haves”, “could-haves” en “won’t haves”. Hierbij is er ook onderscheid gemaakt tussen functionele eisen en niet-functionele eisen. De functionele eisen beschrijven hierbij de daadwerkelijke functies van het product, terwijl de niet-functionele eisen meer over de interactie en het gebruik van het product gaat.

Naast het beschrijven van de eisen en functionaliteiten volgens MoSCoW, zijn er ook acceptatiecriteria opgesteld. Deze acceptatiecriteria beschrijven de minimale eisen die nodig zijn om te beoordelen of het “proof of concept” voldoende is ontwikkeld. Om de acceptatiecriteria op te stellen is er gebruik gemaakt

van de ISO 25010 normering. Dit is een normering voor het beoordelen van softwarekwaliteit. Er is gekozen voor deze normering in plaats van de ISO 9126. Dit is een andere normering voor het beoordelen van softwarekwaliteit. Er is gekozen voor de ISO 25010 omdat dit de herziene versie is van de oudere ISO 9126 normering. In de vernieuwde norm zijn er onder andere extra categorieën toegevoegd op het gebied van beveiliging en compatibiliteit. Deze onderdelen hebben allebei een belangrijke rol in de context van “Password Cracking as a Service”.

Bij het opstellen van de acceptatiecriteria is er ook een risicoanalyse uitgevoerd om mogelijke risico’s in kaart te brengen die ervoor kunnen zorgen dat niet alle acceptatiecriteria worden behaald. Deze risicoanalyse heeft ervoor gezorgd dat er maatregelen konden worden genomen om mogelijke problemen te voorkomen in de ontwikkeling van het “Proof of Concept”.

De daadwerkelijke functionaliteiten, eisen, acceptatiecriteria en risicoanalyse zijn in detail beschreven in de bijlage “*Requirementsanalyse.pdf*”.

7.4 Vergelijkend onderzoek

Voor het vergelijkend onderzoek zijn er verschillende hash-kraakprogramma’s getest op kwaliteit. Voor deze tests zijn de kwaliteitscriteria uit de ISO 25010 gehanteerd. Deze norm omschrijft verschillende aspecten van software die kunnen worden gebruikt om kwaliteit te vergelijken. In het testplan zijn de verschillende kenmerken uit deze software beoordeeld op prioriteit. Aan de hand van deze kenmerken zijn vervolgens tests opgesteld.

De volgende hash-crackingsoftware is getest:

- Hashcat
- JohnTheRipper
- RainbowCrackalack
- PassGAN (i.c.m. hashcat)

Er bestaan meerdere programma’s voor het werken met “rainbow tables”. Een van de meest populaire programma’s hiervoor is “rcrack”. Voor deze software was het echter niet makkelijk om compatibele tabellen te vinden. Het is mogelijk om zelf tabellen te genereren, maar dit kost veel tijd. Uiteindelijk is er gekozen voor een soortgelijk programma, namelijk “rainbowcrackalack”, waarbij wel compatibele tabellen worden meegeleverd.

7.4.1 Prestatie en efficiëntie

Een van de belangrijkste factoren bij het kraken van wachtwoordhashes is de hashrate. Dit is het aantal hashes dat in een bepaalde tijd kan worden getest. Hoe hoger de hashrate, hoe groter de kans dat een hash binnen een bepaalde tijd kan worden gekraakt.

Voor dit onderzoek is er uitgegaan van een enkele “brute-force” aanval op een testdataset van 100 NTLM-hashes. De “brute force” aanval richt zich op karaktercombinaties van 8 karakters lang. De karakterset bevat alle alfanumerieke karakters (A-Z, a-z, 0-9) en een aantal symbolen. In totaal is de karakterset 95 karakters. De totale “keyspace” bestaat dus uit 95^8 karakters. Voor de test met RainbowCrackalack is er gebruik gemaakt van de bestaande meegeleverde “rainbow tables”.

Software	Snelheid (MH/s)	Tijd (uur)	Opmerkingen
Hashcat	27400	65	
JohnTheRipper	19700	93	
RainbowCrackalack	737133	2,5	RainbowCrackalack heeft tijd nodig om een voorberekening uit te voeren. Dit moet per hash apart worden uitgevoerd en wordt niet parallel uitgevoerd. Het uitvoeren van deze berekening kostte op het testsysteem ongeveer **20 seconden per hash**.
PassGAN	n.v.t.	N.v.t	PassGAN is niet in staat om wachtwoorden te kraken, maar kan alleen woordenlijsten genereren voor andere kraakprogramma’s.

Er is een duidelijk snelheidsverschil zichtbaar tussen Hashcat en JohnTheRipper. Hashcat is sneller met een factor van ongeveer **1.39**. Het gebruik van “rainbow tables” in combinatie met RainbowCrackalack is echter nog veel sneller. Het verschil met hashcat is hierbij een factor van ongeveer **26.90**.

7.4.2 Uitwisselbaarheid en toepasbaarheid

Voor het ontwikkelen van de dienst is het belangrijk dat de kraaksoftware kan worden aangestuurd vanuit een ander programma. Voor de dienst maakt de kraaksoftware deel uit van een groter geheel, waardoor het mogelijk moet zijn om de kraaksoftware vanuit een ander programma aan te sturen.

Met deze test wordt er gekeken in welke mate het mogelijk is om elk softwareproduct te automatiseren vanuit de Python-programmeertaal. Hiervoor worden er modules ontwikkeld waarmee de mogelijkheden en beperkingen van elk softwareproduct in kaart kunnen worden gebracht.

Elk programma wordt bij deze test beoordeeld volgens een score die aangeeft in welke mate het mogelijk is om het product aan te sturen vanuit Python. Een hogere score is hierbij een beter resultaat. De scores zijn in de onderstaande tabel beschreven.

Score	Criteria
5	Er is een bestaande library beschikbaar waarmee het programma kan worden aangestuurd. Hiermee is het mogelijk om kraakopdrachten te starten en de uitvoer van het programma te verwerken vanuit Python.
4	Er is geen bestaande library beschikbaar, maar de software kan wel vanuit Python worden gestart en de software kan een uitvoer geven die goed te verwerken is in een automatisch proces (“machine-readable”).
3	Er is geen bestaande library beschikbaar. De software kan vanuit Python worden gestart, maar geeft geen uitvoer die “machine-readable” is. Het is nog wel (deels) mogelijk om met het programma te communiceren en (een deel van) de uitvoer te gebruiken binnen Python.
2	Er is geen bestaande library beschikbaar. De software kan vanuit Python worden gestart, maar geeft geen uitvoer die “machine-readable” is. Het is niet goed mogelijk met het programma te communiceren. De uitvoer van het programma is niet goed te gebruiken binnen Python.
1	De software is alleen aan te sturen via een grafische interface, waardoor er geen praktische manier is om de software vanuit Python te gebruiken.

De resultaten van de tests in de onderstaande tabel beschreven samen met een korte beschrijving van opvallende bevindingen. Een hogere score betekent beter testresultaat.

Software	Score	Opmerking
Hashcat	4	De software kan worden gestart via de subprocess-module in Python. Er is een optie aanwezig om regelmatig de status van het programma als JSON of andere “machine-readable”-formaten weer te geven.
JohnTheRipper	3	De software kan worden gestart via de subprocess-module in Python. De software biedt geen opties voor een uitvoer die geschikt is voor het uitlezen met een ander programma. Het opvragen van de status is specifiek aan het besturingssysteem en systeemconfiguratie. De status van het programma is hierdoor niet goed te controleren. Het is wel mogelijk om gekraakte wachtwoorden uit te lezen.
RainbowCrackalack	2	De software kan worden gestart via de subprocess-module in Python. De software biedt geen opties om de uitvoer te veranderen en de uitvoer is ook niet goed uit te lezen met een programma.
PassGAN	2*	Het programma is een experimentele (proof of concept) Python-module en is niet volwassen. Het programma werkt alleen met een verouderde Python 2. Hierdoor is het programma niet goed te gebruiken vanuit een Python 3 omgeving. *Dit programma genereert alleen woordenlijsten/wachtwoordkandidaten voor het gebruik in andere kraaksoftware en kan niet zelfstandig wachtwoorden kraken.

Uit de tests naar uitwisselbaarheid en toepasbaarheid is gebleken dat hashcat de meeste mogelijkheden biedt om te integreren in diensten zoals “Password Cracking as a Service”. PassGAN en RainbowCrackalack scoren hierbij het slechtst. Er is echter een uitzondering voor PassGAN, omdat dit geen software is voor het kraken van wachtwoorden. Met PassGAN is het mogelijk om woordenlijsten te genereren die te gebruiken zijn in andere kraakprogramma’s. Wanneer een woordenlijst is gegenereerd kan deze constant worden hergebruikt in de andere programma’s, onafhankelijk PassGAN zelf.

7.4.3 Kosten

Het gebruik van software kan gekoppeld zijn aan kosten. Hierbij gaat het niet alleen om kosten voor de aanschaf van de software, maar bijvoorbeeld ook kosten voor het gebruik, het onderhoud, opleidingen, energie en hardware. Het gehele kostenplaatje wordt ook wel de “Total Cost of Ownership” (TCO) genoemd (Gartner, 2018).

Voor het onderzoek naar kosten is er gekeken naar een aantal factoren. Hierbij wordt gewerkt vanuit de aanname dat de organisatie al over de hardware beschikt om de kraaksoftware op uit te voeren. Indien het toch nodig is om extra hardware aan te schaffen voor het gebruik van specifieke software, wordt dit wel meegenomen in de analyse.

Om de geteste software uit te voeren moet er ook een besturingssysteem aanwezig zijn. Aan het gebruik van een besturingssysteem kunnen ook licentiekosten zijn verbonden. Alle geteste software is echter ook beschikbaar voor Linux. Bij elk product wordt er uitgegaan dat er gebruik wordt gemaakt van een gratis Linux-distributie. Hierdoor wordt er in de kostenanalyse uitgegaan dat er geen kosten hoeven worden gemaakt voor het besturingssysteem. Indien de organisatie voor een besturingssysteem kiest waar wel kosten aan zijn verbonden, zoals Microsoft Windows, zijn hier kosten aan verbonden die niet zijn meegerekend in dit onderzoek.

In de onderstaande tabellen zijn de (verwachte) kosten beschreven die bij elk softwareproduct van toepassing kunnen zijn.

7.4.3.1 Aanschaf-/licentiekosten

De onderstaande tabel laat de aanschaf- en licentiekosten zien voor de geteste programma’s.

Software	Kosten	Opmerking
Hashcat	€0	Hashcat is volledig open source en wordt gratis uitgegeven onder de MIT-licentie.
JohnTheRipper	€0/ €185/j (pro)	De pro versie bevat o.a. een woordenlijst van meerdere talen en 1 jaar ondersteuning. JtR is beschikbaar onder de GPLv2 licentie.
RainbowCrackalack	€0	RainbowCrackalack is volledig open source en gratis en wordt uitgegeven onder de GPLv3-licentie.
PassGAN	€0	Passgan is volledig open source en wordt gratis uitgegeven onder de MIT-licentie.

7.4.3.2 Extra hardwarekosten

Er wordt gewerkt vanuit de aanname dat de organisatie al over hardware beschikt voor het gebruiken van de kraaksoftware. Deze analyse beschrijft alleen kosten indien er extra hardware nodig is voor een van de specifieke programma's.

Software	Kosten	Opmerking
Hashcat	€0	Hashcat biedt meer snelheid op betere hardware. Dit is vooral afhankelijk van de GPU(s) van een systeem. Het systeem van de organisatie voldoet hier al aan.
JohnTheRipper	€0	JtR biedt meer snelheid op betere hardware. Dit is vooral afhankelijk van de GPU(s) van een systeem. Het systeem van de organisatie voldoet hier al aan.
RainbowCrackalack	€65-€250	Er is een opslagmedium nodig voor het opslaan van tabellen. Bij de schatting is uitgegaan van NTLM-tabellen van 8 karakters. Hierbij is een opslagmedium nodig van ~1TB. Tabellen kunnen zelf worden gegenereerd of er kan een opslagmedium worden besteld met eerder gegenereerde tabellen. Prijs is vaak afhankelijk van de snelheid van het opslagmedium.
PassGAN	€0	De geteste PassGAN-implementatie werkt alleen wanneer er een GPU in het systeem aanwezig is. Het systeem van de organisatie voldoet hier al aan.

7.4.3.3 Energiekosten

Het kraken van hashes kan veel rekenkracht vragen. Voor deze rekenkracht moet een systeem vaak voor een langere tijd op vol vermogen draaien. Aangezien dit veel stroom kan verbruiken, kan een verschil in efficiëntie tussen programma's een verschil opleveren op de energierekening. Hiernaast heeft een lager energieverbruik ook voordelen op het gebied van milieu.

De tests zijn uitgevoerd op de laptop van de student. Hierdoor is er waarschijnlijk een verschil tussen het geschatte stroomverbruik en het daadwerkelijke stroomverbruik binnen een productieomgeving. Deze analyse moet hierdoor alleen worden gezien als een indicatie van de relatieve verschillen in stroomverbruik van verschillende programma's.

Voor dit onderzoek is er uitgegaan van een enkele "brute-force" aanval op een testdataset van 100 NTLM-hashes. De "brute force" aanval richt zich op karaktercombinaties van 8 karakters lang. De karakterset bevat alle alfanumerieke karakters (A-Z, a-z, 0-9) en een aantal symbolen. In totaal is de karakterset 95 karakters. De totale "keyspace" bestaat dus uit 95^8 karakters.

In de resultaten wordt uitgegaan van een energieprijs van €0.588 per KWh. Dit is de gemiddelde Nederlandse energieprijs van mei 2022 (Overstappen.nl, 2022).

De meest relevante bevindingen is verwerkt in de onderstaande tabel. De data voor hashrate is hierbij hergebruikt van 7.4.1 PRESTATIE EN EFFICIËNTIE. Het totale stroomverbruik is het geschatte stroomverbruik voor het doorlopen van de gehele "keyspace".

Software	Hashrate (MH/s)	Verbruik (W)	Totaal verbruik (KWh)	Kosten (€)
Hashcat	27400	185	12,03	7,07
JohnTheRipper	19700	165	15,34	9,02
RainbowCrackalack	737133	53	0,13	0,08
PassGAN	N.v.t	n.v.t.	n.v.t.	n.v.t.

Uitgebreidere data is beschikbaar in de bijlage "*Stroomkosten.xlsx*".

In de resultaten is duidelijk te zien dat Hashcat ongeveer **1.28** keer zo efficiënt omgaat met de hardware vergeleken met JohnTheRipper en hierdoor ook efficiënter is qua stroomverbruik. Het gebruik van "rainbow tables" in combinatie met RainbowCrackalack is echter nog veel efficiënter in deze test met een factor van **92.54**. Dit is inclusief het stroomverbruik van een 1TB harde schijf. De efficiëntie van "Rainbow Tables" zou in theorie nog meer kunnen toenemen bij het gebruik van een moderne SSD als opslagmedium, aangezien hiermee vaak hogere snelheden kunnen worden en het stroomverbruik nog lager kan liggen.

7.4.4 Forensische correctheid en betrouwbaarheid

Een ander belangrijk onderdeel van software is de betrouwbaarheid en forensische correctheid. De kraaksoftware moet deel uitmaken van een geautomatiseerd proces, waardoor er slechts weinig gebruikersinteractie zal plaatsvinden. Als er binnen een kraakprogramma onverwachte fouten optreden of als de resultaten onbetrouwbaar zijn, kan dit het proces mogelijk verstoren.

Voor de betrouwbaarheid van de programma's kan er onder andere worden gekeken naar de resultaten uit het onderzoek van 7.4.2 UITWISSELBAARHEID EN TOEPASBAARHEID. Indien een programma geen goede ondersteuning heeft voor goede communicatie met andere software, kan dit mogelijk ook voor onvoorspelbaar gedrag zorgen.

Hiernaast is er gekeken naar de forensische correctheid van het kraakproces zelf. Hierbij is er per kraakprogramma een test uitgevoerd met bekende hashes waarvan bekend is dat deze moeten kunnen worden gekraakt. Hierna zijn de resultaten van het kraakprogramma vergeleken met de verwachte resultaten om te controleren of dit overeenkomt. Dit proces is 20 keer herhaald. Indien de resultaten van het kraakprogramma niet overeenkomen met de verwachte resultaten, toont dit aan dat een programma mogelijk onbetrouwbaar is en niet forensisch correct werkt. Er is gekozen voor 20 iteraties omdat dit een goede balans lijkt te geven tussen de betrouwbaarheid van de resultaten en de benodigde tijd.

De resultaten zijn vergeleken door te kijken naar de resulterende “potfile”. Dit is een bestand waarin een kraakprogramma een lijst van gekraakte hashes samen met het bijbehorende wachtwoord bijhoudt.

De resultaten van de tests zijn in de onderstaande tabel beschreven. Hoe hoger het aantal geslaagde iteraties, hoe beter het resultaat.

Software	Uitgevoerde iteraties	Geslaagde iteraties	% geslaagd
Hashcat	20	20	100%
JohnTheRipper	20	20	100%
RainbowCrackalack	20	20	100%
PassGAN	n.v.t.	n.v.t.	n.v.t.

Er zijn geen verschillen zichtbaar in de resultaten van de verschillende kraakprogramma’s. Alle geteste programma’s hebben een score van 100% en er zijn geen onvoorspelbare resultaten gevonden. Dit toont aan dat er tijdens het uitvoeren van de tests geen forensisch incorrect gedrag is vertoond.

Het is met deze testresultaten niet te concluderen of de programma’s ook op een langere termijn forensisch correct blijven werken. De resultaten uit hoofdstuk 7.4.2 UITWISSELBAARHEID EN TOEPASBAARHEID kunnen wel een indicatie geven of er problemen kunnen ontstaan bij de integratie met andere software. Een lagere score kan hierbij een indicatie geven tot een slechter resultaat.

7.5 Ontwikkelen “Proof of Concept”

Het vergelijkend onderzoek resulteert in een advies over wachtwoordkraaksoftware. Dit advies wordt gecombineerd met resultaten uit de overige voorgaande onderzoeken. Om dit advies ook daadwerkelijk te testen en te ondersteunen is er een “Proof of Concept” ontwikkeld waarin de haalbaarheid en uitdagingen van de dienst kunnen worden aangetoond. De functionaliteiten van dit “Proof of Concept” zijn eerder vastgelegd tijdens de requirementanalyse.

Het proof of concept is grofweg onder te verdelen in vier onderdelen:

- Het verkrijgen van wachtwoordhashes uit de bestaande infrastructuur van een klant (Active Directory).
- Het anonimiseren/pseudonimiseren van de wachtwoordhashes en de communicatie met de kraakserver (in beide richtingen).
- Het daadwerkelijke kraken van de wachtwoordhashes.
- Terugkoppeling van de kraakresultaten en automatisch actie ondernemen (bijvoorbeeld wachtwoordreset of vergrendelen account)

Deze verschillende onderdelen worden hieronder in de verschillende subkoppen behandeld.

Om het “Proof of Concept” te ontwikkelen is er gekozen om gebruik te maken van de Python-programmeertaal. Er is hiervoor gekozen door bekendheid van de taal bij de student en de beschikbaarheid van verschillende modules (libraries) voor de interactie met Active Directory. Een ander voordeel van Python is dat er vaak minder code nodig is om een doel te bereiken vergeleken met andere programmeertalen. Een nadeel is echter dat Python vergeleken met andere talen minder snel en efficiënt kan worden uitgevoerd. Snelheid en efficiëntie is binnen de dienst wel een belangrijk onderdeel, maar dit is vooral van toepassing op de software voor het kraken van de wachtwoordhashes zelf. Voor het kraken van de hashes zelf wordt Python niet gebruikt. Hiervoor is er gekozen voor het programma “Hashcat”. De onderbouwing van het gebruik van “Hashcat” is uitgewerkt in hoofdstuk 7.4 VERGELIJKEND ONDERZOEK.

7.5.1 NTLM-hashes uit Active Directory

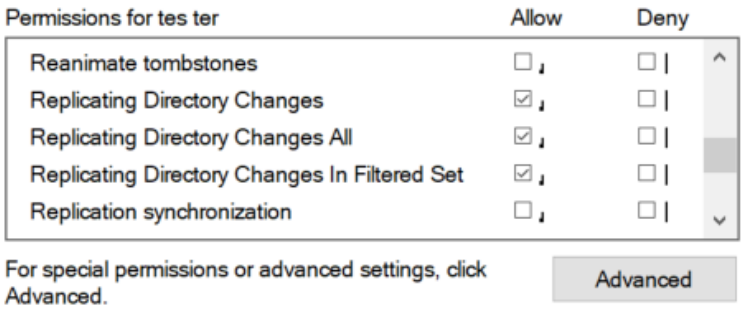
Tijdens het vooronderzoek zijn er verschillende methodes onderzocht om wachtwoordhashes te verkrijgen uit Active Directory. Er is een mogelijkheid om van een externe machine een domain controller na te bootsen en te synchroniseren met de bestaande domain controller. Dit is een zogenaamde “DCSync-aanval”. Een andere methode is om een kopie te maken van het databasebestand (*ntds.dit*) via een zogenaamde “Volume Shadow Copy”. Deze methodes zijn verder beschreven in het literatuuronderzoek.

Uit de requirementanalyse bleek dat er geen specifieke eisen waren voor de methode waarop de wachtwoordhashes werden uitgelezen. Hierom is er methode geselecteerd die vooral het beste lijkt te integreren met Python, aangezien het product hiermee wordt ontwikkeld. Er is uiteindelijk gekozen voor “secretsdump” van impacket. Impacket is een Python-module voor het communiceren met behulp van verschillende netwerkprotocollen. Andere beschikbare oplossingen voor het uitlezen van

wachtwoordhashes maken vooral gebruik van PowerShell, waardoor deze minder geschikt zijn voor het gebruik in Python.

Bij de impacket-module wordt een script genaamd “secretsdump” geleverd waarmee het mogelijk is om een DCSync-aanval uit te voeren. Dit voorbeeldscript is gebruikt om een nieuw soortgelijk script te schrijven met alleen de nodige functionaliteiten. Hierdoor is een script ontstaan van slechts 20% van de grootte van het originele secretsdump script. Dit zorgt voor betere leesbaarheid en doelgerichtheid.

Om toegang te krijgen tot wachtwoordhashes moet er eerst worden ingelogd met een account met de juiste bevoegdheden. De simpelste oplossing is om hiervoor een “Domain Admin” account te gebruiken. Dit is een account met de hoogst mogelijke rechten binnen een Active Directory domein. Volgens het “**Principle of Least Privilege**” mag een onderwerp echter alleen de minimale rechten krijgen die hij nodig heeft (CISA, 2013). Het gebruik van een “Domain Admin” account is hierom tegenstrijdig met dit principe en kan onnodige beveiligingsrisico’s veroorzaken. Om wel volgens het “Principle of Least Privilege” te kunnen werken moet er een apart gebruikersaccount worden gemaakt met alleen de minimale rechten die nodig zijn voor het uitvoeren van DCSync. De rechten die nodig zijn voor DCSync worden in Active Directory beschreven als “Replicating Directory Changes”. Er zijn in totaal drie rechten die in deze categorie vallen. Deze rechten zijn zichtbaar in FIGUUR 2.



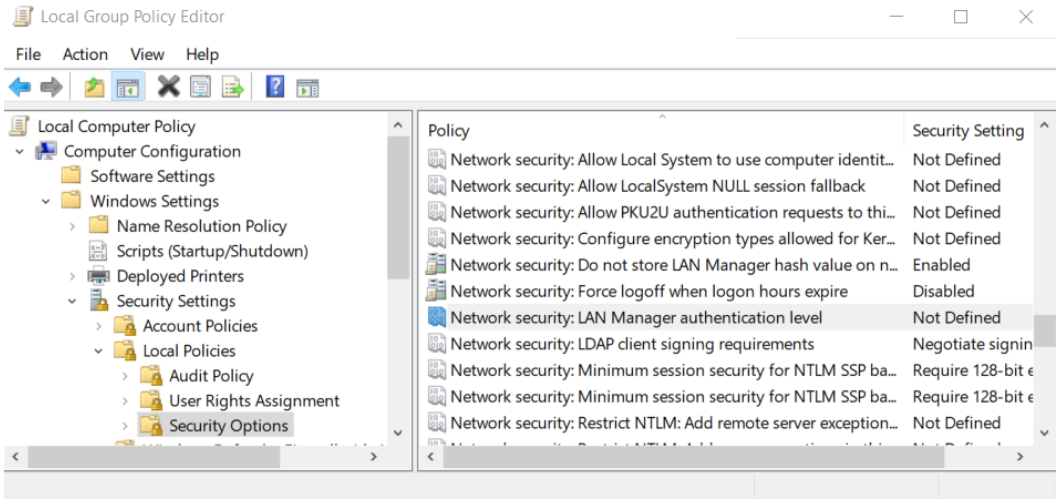
Figuur 2, De drie rechten die nodig zijn voor DCSync

7.5.1.1 Risico’s

Uit verder onderzoek bleek dat het toepassen van de “Replicating Directory Changes”-rechten extra beveiligingsrisico’s kunnen creëren. Met standaardinstellingen van Active Directory is het namelijk toegestaan om alleen met een gebruikersnaam en een wachtwoordhash in te loggen. Hierbij is het originele wachtwoord dus niet nodig.

Het risico dat hierbij ontstaat is dat de “Replicating Directory Changes”-rechten toegang geven tot de hashes van ieder account binnen een domein. Hieronder vallen dus ook administratoraccounts. Op deze manier is het dus mogelijk om indirect toegang te krijgen tot een administratoraccount via het account met rechten voor DCSync.

Dit risico kan worden verholpen door gebruikers toe te voegen aan zogenaamde “Security Groups” of door authenticatie via alleen NTLM-hashes uit te schakelen in een Group Policy FIGUUR 3. Het uitschakelen hiervan kan wel zorgen voor problemen op het gebied van comptabiliteit met oudere versies van Windows (Microsoft, sd). Dit is een afweging waar de afnemer van de dienst zelf een oordeel over zou moeten geven. Microsoft geeft zelf instructies over het voorkomen van deze risico’s op http://go.microsoft.com/fwlink/?LinkId=17936.



Figuur 3, De instelling in Active Directory voor het uitschakelen van authenticatie met alleen NTLM hashes.

7.5.2 Communicatie tussen kraakserver en client

Een belangrijk onderdeel van het “proof of concept” is de communicatie tussen client en kraakserver. Aangezien er hierbij gevoelige gegevens worden overgedragen, is het zeer belangrijk dat deze communicatie veilig verloopt.

Er zijn meerdere methodes onderzocht om de verbinding tussen de twee systemen te beveiligen. Een oplossing hiervoor is het versleutelen van de verbinding tussen de systemen. Dit voorkomt dat een aanvaller gegevens kan buitmaken door de verbinding af te luisteren.

Het versleutelen van een verbinding biedt echter geen bescherming tegen ander soort aanvallen. In het geval dat een aanvaller bijvoorbeeld een kwetsbaarheid vindt in de kraakserver en hierdoor kan inbreken, kan hij toegang krijgen tot gevoelige gegevens van afnemers van de dienst. Om dit risico zo veel mogelijk te verkleinen, is de serverapplicatie ontworpen met het idee dat deze gehackt “mag” worden, zonder dat het mogelijk is om veel vertrouwelijke gegevens buit te maken. Dit betekent niet dat er geen aandacht is voor de overige beveiliging van de kraakserver, maar juist om de beveiliging van de kraakserver in het algemeen te verbeteren.

Vanuit dit oogpunt zijn er ideeën getest om de informatie op de kraakserver zo veel mogelijk te beperken. Het idee hierachter is dat de kraakserver geen informatie bevat die terug te herleiden is naar een specifieke klant of organisatie. Tijdens het literatuuronderzoek zijn er meerdere methodes

onderzocht. Deze methodes richten zich allemaal op het anonimiseren/pseudonimiseren van de gegevens die worden overgedragen naar de kraakserver. Deze verschillende methodes zijn hieronder samen met de voor- en nadelen kort beschreven. In het literatuuronderzoek is een meer uitgebreide beschrijving beschikbaar over de werking van elk van deze methodes.

7.5.2.1 Methodes

Pseudonimiseren met unieke id (uuid)

Bij deze methode genereert de clientapplicatie een willekeurig gegenereerd uniek id en stuurt alleen dit id, samen met de wachtwoordhashes, naar de kraakserver. De kraakserver bewaart vervolgens alleen het id en de wachtwoordhashes en slaat dus geen informatie op over de afkomst van het verzoek.

Het voordeel van deze methode is dat de kraakserver hierdoor met slechts heel weinig informatie kan werken en geen informatie hoeft te bewaren over de identiteit van de klant. Indien een aanvaller toegang krijgt tot de kraakserver, zal hij alleen maar een willekeurige reeks aan letters en cijfers kunnen zien, waardoor het erg moeilijk wordt om dit terug te herleiden naar een specifieke afnemer van de dienst. Een nadeel is echter dat de kraakserver zelf geen resultaten kan terugkoppelen aan de client. De clientapplicatie zal zelf regelmatig moeten opvragen wat de status is van een specifieke taak.

K-anonymity

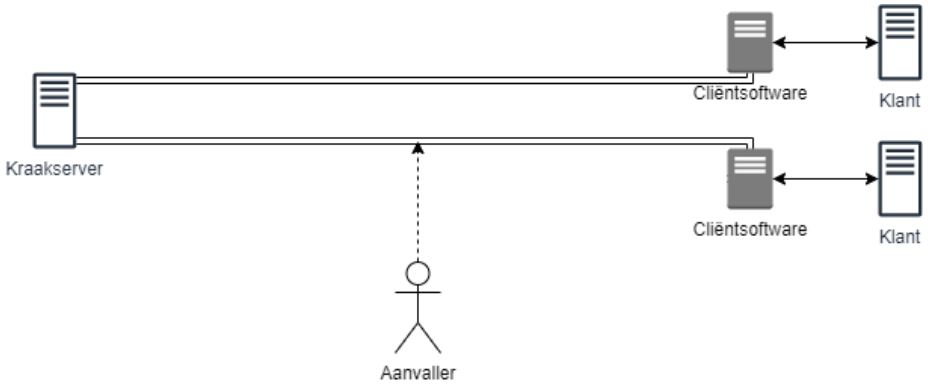
K-anonymity is een techniek die kan worden toegepast bij het opvragen van de status van een specifieke wachtwoordhash. Hierbij “knipt” de client de hash af en doet een verzoek bij de kraakserver om alle zwakke hashes door te geven die met het afgeknipte deel beginnen. De kraakserver geeft hierbij een groot antwoord van meerdere zwakke hashes. De client kan vervolgens controleren of de originele wachtwoordhash in deze lijst voorkomt en dus als zwak moet worden beschouwd.

Een voordeel van deze methode is dat het niet meer nodig is om hashes te groeperen per klant, aangezien alleen de informatie over een specifieke hash wordt opgrevraagd. Een nadeel van deze methode is echter dat de client voor elk gebruikersaccount een apart verzoek moet doen. Dit kan voor veel vertragingen zorgen die vooral merkbaar zal zijn bij grotere organisaties met veel gebruikersaccounts. Ook is deze methode alleen effectief als de kraakserver over een grote hoeveelheid hashes beschikt om in het antwoord naar de client te verwerken. Deze methode biedt dus verschillende uitdagingen bij inzet op kleine en grote schaal.

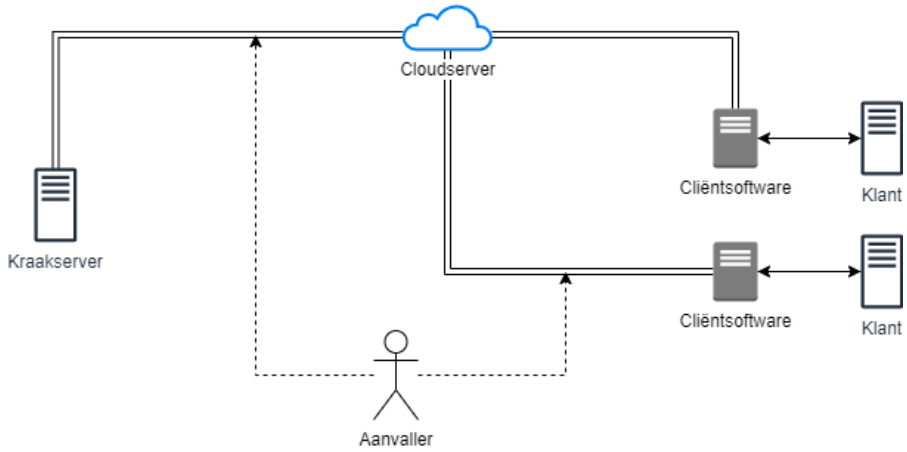
“Tussennodes”

Bij het gebruik van “tussennodes” zullen kraakserver en client niet direct met elkaar communiceren, maar enkel via een tussenpersoon. Denk hierbij bijvoorbeeld aan gedeelde cloudopslag. Als een aanvaller toch toegang krijgt tot de server of de client, zal hij alleen communicatie kunnen zien met deze gedeelde opslag. Hierdoor wordt er geen informatie vrijgegeven over de andere systemen die ook alleen maar via de gedeelde cloudopslag communiceren. (Zie FIGUUR 4, FIGUUR 5) De daadwerkelijke data moet dan ook worden beschermd door middel van encryptie, zodat deze ook niet kan worden gelezen door de tussenpersoon of andere partijen die toegang hebben tot de gedeelde ruimte.

Een voordeel van deze methode is dat het veel moeilijker wordt om informatie over afnemers van de dienst te verkrijgen door het afluisteren van de verbinding. Een nadeel is echter dat er veel extra infrastructuur nodig is en de dienst nu afhankelijk is van een externe partij. Ook moet de tussenpersoon kunnen worden vertrouwd, aangezien deze toegang krijgt tot informatie van alle partijen.



Figuur 4, Voorbeeld zonder "tussennodes". De aanvaller ziet de verbinding tussen kraakserver en de cliënt



Figuur 5, Voorbeeld met "tussennode". De aanvaller ziet alleen verkeer met de cloudserver en geen directe verbinding tussen kraakserver en klant.

7.5.2.2 Keuzes

Voor het "proof of concept" is er een afweging gemaakt tussen de bovenstaande methodes op basis van hun voor- en nadelen. Uiteindelijk is er een soort "hybrideoplossing" ontstaan waarbij verschillende voordelen worden gecombineerd.

Ten eerste is er gekozen om alle communicatie tussen kraakserver en client te versleutelen. Hierbij is er gekozen om te communiceren via "https". Hiervoor is gekozen omdat dit makkelijk te implementeren is

als laag op een niet-versleutelde “http”-verbinding en omdat dit de clientapplicatie ook meteen de mogelijkheid geeft om de identiteit van de kraakserver te verifiëren.

Verder is er gekozen voor het gebruiken van een unieke id (uuid) per taak. Dit zorgt ervoor dat de kraakserver geen informatie hoeft bij te houden over de identiteit van verschillende klanten, maar toch hashes kan groeperen op een manier waarbij de resultaten van een “kraaktest”

Een beperking die ontstaat door het pseudonimiseren van klanten, is dat het niet meer goed mogelijk is om na te gaan welke taak bij welke klant hoort. Dit maakt het in theorie voor iedereen mogelijk om de status van een taak op te vragen bij de kraakserver, mits die persoon kennis heeft over het gebruikte “uuid” van de taak. De kans dat een aanvaller een uuid kan raden is extreem klein, omdat dit een lange reeks aan willekeurig gegenereerde karakters is. Dit lijkt relatief veilig, maar kan worden beschreven als “**Security Through Obscurity**”. Dit wordt door onder andere het NIST afgeraden en moet op zichzelf niet worden gezien als een goede veiligheidsmaatregel (NIST, 2008).

Dit heeft geleid tot een oplossing die lijkt op K-anonymity, maar toch goed in combinatie met de unieke ids kan worden gebruikt. Bij deze oplossing kan de client de status van een taak opvragen via een “uuid”, maar krijgt hierbij alleen afgeknipte hashes terug in het antwoord. De kraakserver zorgt ervoor dat er zo veel mogelijk van de hash wordt afgeknipt, zonder dat er conflicten ontstaan met dubbele waarden. De afgeknipte hashes zijn hierdoor alleen bruikbaar voor iemand die de volledige hashes al heeft. Dit is in dit geval alleen de client. Een aanvaller die de status van een “kraaktaak” weet op te vragen krijgt dus alleen kleine stukjes van zwakke hashes. Deze zijn in de praktijk niet meer terug te herleiden naar de originele hashes.

```
// 20220415144948
// http://localhost:8080/task_status?uuid=3a8d0ce9-a

{
  "uuid": "3a8d0ce9-a9a8-4a55-80cd-06293d124df9",
  "is_done": 0,
  "weak_short_hashes": [
    "fc",
    "31",
    "1c",
    "3b",
    "0d",
    "39",
    "db"
  ]
}
```

Figuur 6, Een antwoord van de kraakserver over de status van een taak. De “afgeknipte” hashes zijn zichtbaar.

7.5.3 Kraken van wachtwoordhashes

Voor het kraken van hashes is er gebruik gemaakt van “hashcat”. Uit het vergelijkend onderzoek is gebleken dat dit de meest geschikte software is voor het project. Hashcat is volledig gratis te gebruiken,

bleek het snelst en meest efficiënt te werken vergeleken met de andere programma's. Ook biedt hashcat opties om uitvoer te genereren in een vorm die makkelijk uit te lezen is binnen een script ("machine-readable"). Dit is verder beschreven in het hoofdstuk 7.4 VERGELIJKEND ONDERZOEK.

7.5.3.1 Benchmarks

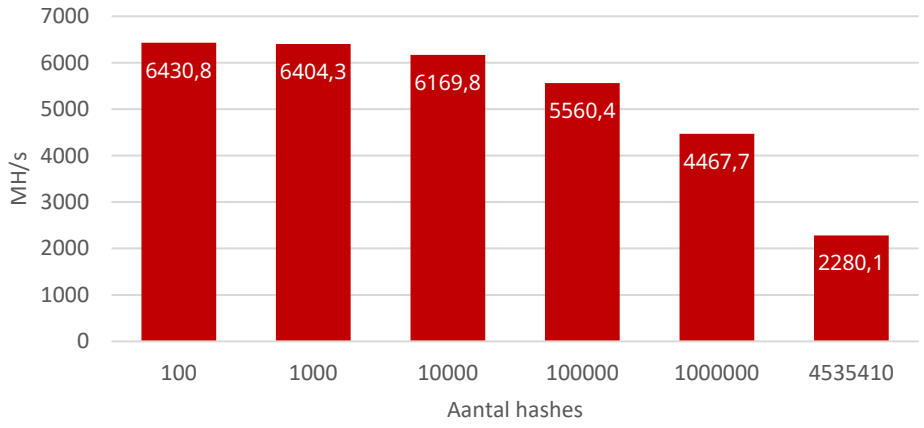
Vanuit de stageorganisatie was ook de vraag of er een verband is tussen de snelheid van hashcat en het aantal hashes dat tegelijkertijd wordt getest. Hierbij was de verwachting dat hashcat (en andere kraaksoftware) efficiënter is bij het kraken van meerdere hashes tegelijk.

Om deze theorie te testen is er een benchmark uitgevoerd waarbij de snelheid van hashcat met verschillende hoeveelheden hashes is getest. Vervolgens is er gekeken naar het effect dat dit aantal heeft op de benodigde tijd om een kraaktest uit te voeren.

De testdata is afkomstig uit de "Compilation Of Many Breaches" (COMB). Dit is een verzameling van accountgegevens afkomstig uit verschillende datalekken. Om de testdata op te stellen is er gekozen om de COMB te filteren op alleen .nl e-mailadressen. Hiervoor is gekozen omdat de stageorganisatie voornamelijk Nederlandse klanten heeft en het gebruik van testdata op basis van Nederlandse accounts mogelijk realistischere resultaten geeft. Door middel van een Python-script is er gezocht op deze e-mailadressen en vervolgens is het bijbehorende wachtwoord gehasht met NTLM. Dit resulteerde in een lijst van 4.535.410 hashes. Voor de benchmarks is vervolgens steeds een subset van deze lijst gebruikt.

Bij elke test is er gekozen voor een "dictionary + rule attack". Hierbij is er gebruik gemaakt van Nederlandse woordenlijsten die zijn aangeleverd door de stageorganisatie. Hiernaast is er gebruik gemaakt van de "dive ruleset". Dit is de meest uitgebreide set aan regels die met Hashcat wordt meegeleverd. Het aantal geteste hashes is steeds vermenigvuldigd met 10, startend bij 100 en eindigend bij de volledige lijst van 4.535.410.

De resultaten van de tests zijn uitgewerkt in FIGUUR 7. De snelheid wordt weergegeven als de "hashrate" in MH/s. Dit is het aantal miljoenen hashes dat per seconde wordt getest. Hoe hoger de hashrate hoe beter het resultaat.



Figuur 7, Hashrate vs. Aantal geteste hashes

Tijdens het uitvoeren van het onderzoek viel op dat de hashrate ook negatief werd beïnvloed door het aantal hashes die door hashcat waren gevonden. Van de testdata afkomstig uit de COMB werd bij de tests steeds rond de 94% van de wachtwoorden gekraakt. Bij de grootste test van 4.535.410 hashes werden er 4.275.002 hashes daadwerkelijk gekraakt. Bij deze test was de gemiddelde hashrate **2280,1 MH/s**. Om te testen wat de invloed is van het vinden van veel hashes, is er ook een test uitgevoerd met een set nephashes van dezelfde grootte. De gemiddelde hashrate bij deze test was **4113,9 MH/s**. Dit toont aan dat de snelheid van hashcat bij de eerdere tests werd beïnvloed door het grote aantal gekraakte hashes en in andere scenario's een hogere hashrate zou kunnen hebben.

In de resultaten is een verband zichtbaar tussen het aantal geteste hashes en de hashrate. Het diagram laat zien dat het aantal geteste hashes een negatief invloed heeft op de hashrate, maar dit geen lineair verband is. Het verschil tussen 1,000 en 10,000 hashes is een toename van **900%**. De hashrate gaat echter met slechts **~4%** omlaag. Dit toont aan dat het veel efficiënter is om veel hashes te verzamelen en vervolgens tegelijkertijd te testen, in plaats van kleine hoeveelheden hashes los van elkaar te testen. Deze tests laten zien dat een centrale kraakserver veel efficiënter is qua snelheid dan een aparte kraakmachine per klant. Door het gebruik van een centrale kraakserver kan er dus veel efficiënter worden omgegaan met hardware.

De ruwe testdata is beschikbaar in de bijlage "hashcat benchmark.xlsx"

7.5.4 Automatische acties

Een van de wensen uit de requirementanalyse is dat er automatisch actie kan worden ondernomen aan de hand van de resultaten van een kraaktest. Zo is het bijvoorbeeld mogelijk om automatisch een wachtwoordreset af te dwingen of een account te vergrendelen wanneer er een wachtwoord van een gebruiker is gekraakt.

7.5.4.1 Techniek

Er zijn verschillende methodes onderzocht waarmee het mogelijk is om automatisch acties te ondernemen op gebruikers binnen Active Directory. Veel functies van Active Directory kunnen worden aangestuurd via PowerShell, waaronder ook het vergrendelen van accounts of het afdwingen van een automatische wachtwoordreset. Zoals ook werd aangegeven in hoofdstuk 7.5.1 NTLM-HASHES UIT ACTIVE DIRECTORY, is het gebruik van PowerShell niet ideaal, omdat de rest van het "proof of concept" in Python wordt ontwikkeld. Een alternatief hiervoor is het Lightweight Directory Access Protocol (LDAP). Dit is een protocol waarmee het mogelijk is om informatie op te vragen en aanpassingen te doen in verschillende directory-diensten, waaronder ook Active Directory. Er zijn ook Python-modules beschikbaar voor het werken met LDAP (ldap3), waardoor dit de meest geschikte keuze leek.

Door gebruik te maken van LDAP zijn er functies ontwikkeld waarmee het mogelijk is om automatisch een wachtwoord reset af te dwingen voor geselecteerde gebruikers. Wanneer deze functie wordt gebruikt, moet de gebruiker bij de volgende inlogpoging een nieuw wachtwoord kiezen. Hiernaast is ook de functie toegevoegd om een account volledig te uit te schakelen. Dit account is dan vergrendeld totdat een beheerder het account weer ontgrendeld. Dit kan bijvoorbeeld worden ingezet bij accounts die toegang hebben tot zeer gevoelige gegevens of wanneer een gebruiker niet binnen een bepaalde

tijd zijn wachtwoord heeft gereset. Wanneer deze automatische acties worden toegepast is afhankelijk van de wensen van de afnemer van de dienst. Hierom is er gekozen om het mogelijk te maken om deze functies in- en uit te schakelen. Voor testdoeleinden zijn er ook functies ontwikkeld waarmee deze veranderingen ongedaan kunnen worden gemaakt, zoals het ontgrendelen van een account en het ongedaan maken van een gedwongen wachtwoordreset.

Tijdens het ontwikkelen van deze functies bleek dat sommige LDAP-functies soms onvoorspelbaar werken. Het instellen van een wachtwoordreset werkt in Active Directory bijvoorbeeld door het wachtwoord te laten verlopen. Een van de problemen die hierbij kunnen ontstaan is dat het niet zomaar mogelijk is om een wachtwoordreset af te dwingen bij een gebruikersaccount waarvan is ingesteld dat het wachtwoord nooit verloopt. Bij het aanroepen van de LDAP-functie geeft Active Directory hierbij echter wel het “success”-antwoord, wat het idee zou geven dat de verandering succesvol is toegepast, ook al is dit niet gebeurd. Dit gedrag lijkt onvoorspelbaar en is daardoor niet “forensisch correct”. Om dit probleem op te lossen is er gekozen om na het gebruik van elke LDAP-functie de huidige status van het account op te vragen en te controleren. Hierbij wordt er gecontroleerd of de verandering daadwerkelijk is toegepast. Wanneer er wordt bevonden dat dit niet het geval is, zal de functie falen en hier een foutmelding over geven. Hierdoor kan onvoorspelbaar gedrag worden gedetecteerd en afgevangen, waardoor de “forensische correctheid” wordt verbeterd.

7.5.4.2 Functies

Tijdens de requirementanalyse is er door middel van de MoSCoW-methode een lijst van mogelijke automatische acties opgesteld. Hieronder wordt een korte beschrijving gegeven van de ontwikkelde functies.

- **Automatische wachtwoordreset:** het programma kan automatisch een wachtwoordreset afdwingen van een account waarbij het wachtwoord is gekraakt. In Active Directory kan dit worden gedaan door de “*pwdLastSet*” van een gebruiker op 0 te zetten. Het programma kan voor testdoeleinden ook een automatische wachtwoordreset ongedaan maken door de “pwdLastSet” waarde weer op -1 te zetten. Het afdwingen van een wachtwoordreset is niet mogelijk wanneer een gebruiker een wachtwoord heeft wat nooit verloopt. Dit is een beperking in Active Directory. Het programma houdt hier rekening mee en kan een foutmelding geven wanneer dit toch wordt geprobeerd.

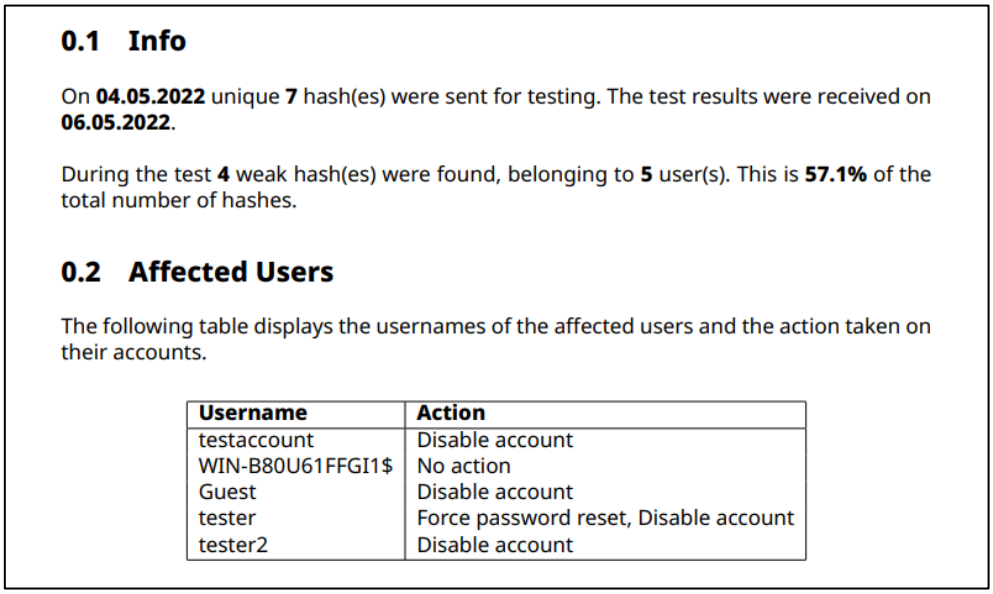
  Om het mogelijk te maken om een automatische wachtwoordreset in te stellen, moeten er in Active Directory een aantal minimale rechten worden ingesteld. Het account dat wordt gebruikt moet minimaal de rechten hebben voor “*Reset user password and force password change at next logon*”. Dit geeft onder andere toegang tot het lezen en schrijven van het “*pwdLastSet*”-attribuut voor “*User*”-objecten. De wijze waarop deze rechten kunnen worden ingesteld staat verder beschreven in de bijlage “*Handleiding PoC.pdf*”.
- **Account vergrendelen:** het programma kan automatisch een account vergrendelen wanneer het wachtwoord van dit account wordt gekraakt. Dit houdt in dat een account volledig wordt uitgeschakeld totdat een beheerder het account weer ontgrendelt. In Active Directory kan een account worden vergrendeld door de “*ACCOUNT_DISABLE*”-bit in de “*UserAccountControl*” bit flags in

te schakelen. Voor testdoeleinden is het ook mogelijk om via het programma een account weer te ontgrendelen.

Om het mogelijk te maken om accounts te vergrendelen, moeten er in Active Directory een aantal minimale rechten worden ingesteld. Het account dat wordt gebruikt voor de cliëntapplicatie moet minimaal de rechten hebben om voor het lezen en schrijven van het "*UserAccountControl*"-attribuut voor *User*-objecten. De wijze waarop deze rechten kunnen worden ingesteld staat verder beschreven in de bijlage "*Handleiding PoC.pdf*".

- **Uitsluiten van computer- en serviceaccounts:** Het programma kan worden geconfigureerd om computer- en serviceaccounts niet te vergrendelen. Deze accounts zullen nog wel worden weergegeven in het testrapport, zodat kwetsbare accounts niet onzichtbaar blijven. Hierdoor kan er bijvoorbeeld worden voorkomen dat bepaalde diensten niet onverwachts worden stilgelegd. Er is binnen Active Directory geen expliciete methode om te controleren of een account een computer- of serviceaccount is. Het programma kijkt hiervoor naar de gebruikersnaam van een account. Wanneer de gebruikersnaam van een account eindigt met "*$*", zal het account als computeraccount worden gezien. Indien een gebruikersnaam begint met "*svc_*", "*service_*", "*serviceaccount_*" of "*sa_*", zal het account als een serviceaccount worden gezien. De daadwerkelijke configuratie kan per organisatie verschillen, waardoor dit waarschijnlijk per klant moet worden besproken.
- **Testrapport:** Het programma kan een rapport genereren met de resultaten van een kraaktest. Het doel van dit rapport is om een korte samenvatting weer te geven van de resultaten van de kraaktest. Het rapport wordt gegenereerd door middel van LaTeX. Dit zorgt ervoor dat er een simpel sjabloon als tekstbestand kan worden gemaakt, dat vervolgens kan worden ingevuld door het programma. Voor het invullen van het sjabloon wordt er gebruik gemaakt van de Python-library "jinja2", een zogenaamde "templating engine". Dit ingevulde sjabloon wordt vervolgens verwerkt tot een pdf-bestand. Een voorbeeld van een rapport is zichtbaar in FIGUUR 8. Het rapport bevat informatie over de datum van de kraaktest, het aantal geteste hashes, het aantal gekraakte hashes en het aantal gekraakte gebruikersaccounts. Hiernaast is er een apart hoofdstuk met de gebruikersnamen van de aangetroffen zwakke accounts en de acties die automatisch op deze accounts zijn toegepast. Indien het niet is gelukt om een actie toe te passen op een account, is dit ook zichtbaar in deze tabel.
- **Organisational Units:** In de requirementanalyse zijn er functies beschreven waarbij er een verschillend beleid kan worden toegepast per "Organisational Unit" (OU) binnen Active Directory. Zo is het mogelijk om bijvoorbeeld een strenger beleid toe te passen voor specifieke afdelingen binnen een organisatie. Deze functie is niet volledig ontwikkeld, maar er is wel een functie ontwikkeld in het programma om informatie te verkrijgen van het OU van een gebruikersaccount. Dit toont aan dat het mogelijk is om op basis van OU te handelen en deze functie in de toekomst kan worden uitgebouwd.
- **E-mail**: In de requirementanalyse is ook een functie beschreven voor het informeren van een gebruiker met een e-mailbericht. Zo kan een gebruiker worden geïnformeerd wanneer het gelukt is om het wachtwoord van de gebruiker te kraken. Deze functie is niet volledig ontwikkeld, maar er is wel een functie ontwikkeld om het e-mailadres van een gebruiker te verkrijgen uit Active Directory. Ook beschikt Python over een ingebouwde e-mail library waarmee het mogelijk is om e-mailberichten te versturen. Dit toont aan dat het mogelijk is om automatisch een e-mailbericht te

sturen naar een gebruiker, mits er een e-mailadres is vastgelegd in Active Directory. Deze functie kan in de toekomst verder worden uitgebouwd.



0.1 Info

On **04.05.2022** unique **7** hash(es) were sent for testing. The test results were received on **06.05.2022**.

During the test **4** weak hash(es) were found, belonging to **5** user(s). This is **57.1%** of the total number of hashes.

0.2 Affected Users

The following table displays the usernames of the affected users and the action taken on their accounts.

Username	Action
testaccount	Disable account
WIN-B80U61FFGI1$	No action
Guest	Disable account
tester	Force password reset, Disable account
tester2	Disable account

Figuur 8, Een automatisch gegenereerd rapport van een kraaktest.

Alle verschillende functionaliteiten van het programma zijn tot slot getest volgens het bijgeleverde testplan. Het testplan is meegeleverd in bijlage "*Testplan PoC.pdf*". Tijdens het uitvoeren van de tests zijn er geen bijzonderheden gevonden. De resultaten van de testcases zijn beschreven in bijlage "*Testcases PoC.xlsx*".

8 Resultaten

In dit hoofdstuk worden de resultaten van de gehele afstudeeropdracht samengevat. Dit beschrijft de ontwikkelde producten en een terugkoppeling op de functies die zijn ontwikkeld voor het “proof of concept”.

In het literatuuronderzoek is er onderzoek gedaan naar informatiebeveiliging en trends op het gebied van ICT-infrastructuur in grote organisaties en de beveiliging hiervan. Op basis van hiervan is besloten om de rest van het project te richten op het kraken van wachtwoordhashes uit Active Directory.

Er is een testomgeving opgezet om de infrastructuur van de dienst na te bootsen. Door onderzoek naar efficiëntie van kraakprogramma’s en technieken om veilig te kunnen communiceren is geconcludeerd dat oplossing met een enkele kraakserver en meerdere cliënten de meest efficiënte oplossing is. De testomgeving is gebaseerd op inbreng van eigen onderzoek en van inbreng van verschillende stakeholders afkomstig uit gehouden interviews. De testomgeving bestaat uit drie virtual machines: een Windows Server systeem met Active Directory om de “domain controller“ van de klant na te bootsen, een Ubuntu Server systeem die dient als clientapplicatie en een tweede Ubuntu Server systeem die dient als centrale kraakserver. De keuzes hiervoor zijn beschreven in bijlages *“Literatuuronderzoek.pdf”, “Requirementsanalyse.pdf” en ”Adviesrapportage.pdf”*. De handleiding voor het opstellen van de testomgeving is beschikbaar in bijlage “Installatiehandleiding testomgeving.pdf”.

Vervolgens is er onderzoek uitgevoerd naar verschillende softwareoplossingen voor het kraken van wachtwoorden. Hierbij zijn er softwaretests uitgevoerd op basis van de kwaliteitscriteria afkomstig uit de ISO 25010. Tijdens de tests zijn verschillende programma’s vergeleken op het gebied van prestaties, efficiëntie, uitwisselbaarheid, toepasbaarheid en kosten. Ook maken de geteste programma’s soms gebruik van verschillende technieken om hashes te kraken. Zo geven de uitgevoerde tests niet alleen informatie over de efficiëntie van software, maar ook de verschillen in de gebruikte technieken. Dit heeft geresulteerd in een advies over het kiezen van een kraakprogramma met hierbij ook een advies over een aanpak voor het kraken van wachtwoorden. Aan de hand van de testresultaten is er uiteindelijk geadviseerd om gebruik te maken van “hashcat” als primair kraakprogramma. Verder kan dit in specifieke gevallen worden ondersteund door een kraakprogramma die gebruik maakt van “rainbow tables”. In de testresultaten is dit aangetoond met het programma “RainbowCrackalack”. De uitgevoerde tests zijn beschreven in bijlage “*Testplan.pdf*” en “*Adviesrapportage.pdf*”. De inbreng vanuit de interviews de MoSCoW prioritering hiervan zijn beschreven in bijlage “*Requirementsanalyse.pdf*”.

Hiernaast is er een “proof of concept” gebouwd om de verschillende adviezen mee te ondersteunen. In dit “proof of concept” zijn de volgende onderdelen ontwikkeld:

Must haves:

- Het programma kan wachtwoordhashes verzamelen uit Active Directory.
- Het programma kan de verkregen gebruikersinfo pseudonimiseren.
- Het programma kan (geanonimiseerde) wachtwoordhashes versturen naar de kraakserver.
- Het programma kan resultaten van een kraaktest van de kraakserver opvragen.
- Het programma kan wachtwoordhashes kraken door middel van hashcat of andere hash-kraaksoftware.
- Het programma kan de resultaten van een kraaktest beschikbaar stellen voor de clientapplicatie.
- De programma's kunnen op een veilige manier communiceren met elkaar en het domein van een klant.
- De programma's werken forensisch betrouwbaar en vertonen geen onvoorspelbaar fout gedrag.
- Het programma bewaart/verwerkt alleen nodige gegevens en voorkomt het opslaan/verwerken van onnodige gegevens. (Bijvoorbeeld geen gebruikersnamen of wachtwoorden als platte tekst).
- De programma's maken alleen gebruik van de minimale benodigde bevoegdheden. (Principle of Least Privilege)

Should haves:

- Het programma kan met meerdere clients communiceren.
- Het programma versleutelt de communicatie tussen serverapplicatie en clientapplicatie.
- Het programma anonimiseert de communicatie tussen serverapplicatie en clientapplicatie.
- Het programma kan automatisch accounts in Active Directory vergrendelen op basis van resultaten van een kraaktest.
- Het programma op basis van resultaten van een kraaktest een wachtwoordreset afdwingen bij gebruikers in Active Directory.
- Het programma kan worden ingezet in organisaties met meerdere domeinen.
- Het programma werkt (bijna) volledig automatisch en er is nauwelijks of geen gebruikersinteractie nodig voor het functioneren van het programma.

Could haves:

- Het programma kan een rapport met statistieken genereren met resultaten van kraaktests.
- (Gedeeltelijk) Het gedrag van het programma kan worden geconfigureerd per "Organisational Unit" (OU).
- (Gedeeltelijk) Het programma kan via e-mail een melding naar een gebruiker versturen wanneer zijn/haar wachtwoord gekraakt is.

De programma's voldoen met deze functies aan alle acceptatiecriteria zoals beschreven in de requirementanalyse. Bij het product zijn installatie- en gebruikershandleidingen beschikbaar. Verder beschrijft het adviesrapport de mogelijke risico's die kunnen ontstaan bij het gebruik van het product, samen met een advies voor het voorkomen hiervan.

9 Conclusie

Dit hoofdstuk beschrijft de conclusie van de afstudeeropdracht. In hoofdstuk 4.1.1 Hoofd-/Deelvragen zijn er verschillende onderzoeksvragen opgesteld die richting hebben gegeven aan de afstudeeropdracht. In dit hoofdstuk worden deze hoofd- en deelvragen concreet beantwoord. Hiernaast wordt er bij sommige deelvragen ook extra inzichten of mogelijkheden tot vervolgonderzoek genoemd.

Wat zijn “zwakke” wachtwoorden?

Een zwak wachtwoord is een wachtwoord dat makkelijk kan worden achterhaald door een aanvaller. Een wachtwoord kan makkelijk worden achterhaald omdat het niet complex genoeg is. Een wachtwoord is meestal niet complex genoeg wanneer het 8 karakters of minder bevat of omdat er veelgebruikte patronen in worden gebruikt (“123456”, “qwerty”, etc.). Er zijn ook regelmatig datalekken waarbij wachtwoorden worden buitgemaakt. Wanneer dit gebeurt, moet een wachtwoord ook als zwak worden beschouwd, omdat er hier een risico is op “credential stuffing”. Een wachtwoord dat wordt hergebruikt voor verschillende diensten kan hierdoor ook als zwak worden beschouwd.

Bij veel diensten wordt er een extra beveiligingsstap toegevoegd in de vorm van multi-factorauthenticatie. Hierbij volstaat een wachtwoord op zichzelf niet en wordt er ook gebruik gemaakt van een extra factor, zoals een biometrische controle of een controle op het bezit van een specifiek voorwerp (telefoon of hardwaresleutel). Doordat deze extra stap soms invloed heeft op de gebruiksvriendelijkheid, kan dit juist extra beveiligingsrisico’s met zich meebrengen. Mensen kunnen hierdoor geneigd zijn om een zwakker wachtwoord te kiezen, omdat ze zo tijd en moeite kunnen besparen bij het inloggen. Er zijn dus veel factoren die kunnen leiden tot het gebruik van zwakke wachtwoorden, zelfs als een organisatie denkt een sterk beleid te hebben.

Op welke wijze kunnen zwakke wachtwoorden worden misbruikt/aangevallen?

Er zijn meerdere aanvallen mogelijk die zich richten op zwakke wachtwoorden. Een methode hiervan is “credential stuffing”, waarbij wachtwoorden van een datalek worden uitgeprobeerd bij andere diensten. Dit kan succesvol zijn wanneer een gebruiker een wachtwoord meerdere keren gebruikt voor verschillende diensten. Een andere methode is “hash cracking”. Deze methode wordt in een volgende deelvraag verder beschreven.

Ook worden er veel wachtwoorden buitgemaakt door middel van “phishing”. Hierbij wordt er bijvoorbeeld gebruik gemaakt van nepmails en vervalste inlogpagina’s. Deze methode maakt echter geen misbruik van zwakke wachtwoorden, maar speelt vooral in op het vertrouwen van een gebruiker. Methodes zoals “phishing” geven aan dat het voorkomen van zwakke wachtwoorden niet altijd een volledige oplossing is. Het testen van de sterkte van een wachtwoord biedt hier ook geen oplossing voor, aangezien deze methodes zich meer richten op het misbruiken van de goedgelovigheid van een gebruiker. Dit toont aan dat wachtwoordbeveiliging een combinatie is van veel verschillende factoren en het testen van de “kraakbaarheid” van wachtwoorden een toevoeging kan zijn voor het verbeteren van wachtwoordbeveiliging in een groter geheel.

Hoe kan een organisatie zich beschermen tegen zwakke wachtwoorden?

Een organisatie kan zich beschermen tegen het gebruik van zwakke wachtwoorden door een goed wachtwoordbeleid op te stellen. Een goed wachtwoordbeleid stelt eisen aan de complexiteit van wachtwoorden, zodat methodes zoals “hash cracking” moeilijker worden. Verder speelt ook gebruiksvriendelijkheid een grote rol voor veel medewerkers. Indien een beveiligingsbeleid ten koste gaat van gebruiksvriendelijkheid, bijvoorbeeld bij multi-factorauthenticatie, is het mogelijk dat medewerkers een beveiligingsbeleid niet willen opvolgen. Ook de infrastructuur van de klant zelf speelt hierbij een rol, omdat wachtwoorden op een juiste manier moeten worden opgeslagen, zoals met een sterk hashingalgoritme dat geschikt is voor gebruik met wachtwoorden. Een manier om de effectiviteit van een wachtwoordbeleid te testen is om een poging te doen om wachtwoorden van medewerkers te kraken. Dit wordt mogelijk gemaakt met een dienst zoals “Password Cracking as a Service”.

Wat is “hash cracking”?

Bij “hash cracking” aanval gedaan op gehashte wachtwoorden. Bij het kraken van hashes wordt er geprobeerd om een wachtwoordhash terug te draaien naar de oorspronkelijke vorm, zodat het originele wachtwoord zichtbaar wordt. Er zijn verschillende manieren om hashes te kraken. Deze methodes worden verder beschreven in een volgende deelvraag.

“Hash cracking” is vooral succesvol op wachtwoorden op zwakke wachtwoorden die op een ongeschikte manier zijn opgeslagen. Hier wordt in andere deelvragen verder op ingegaan.

Hoe vormt “hash cracking” een dreiging voor de beveiliging van organisaties?

“Hash cracking” maakt het mogelijk om zwakke wachtwoorden te onthullen, ook al zijn deze gehasht opgeslagen. Een aanvaller moet wel eerst over de wachtwoordhashes beschikken om deze te proberen te kraken. Het is echter ook mogelijk dat er wachtwoordhashes worden buitgemaakt van een datalek bij een andere organisatie. Als deze hashes worden gekraakt en er zijn gebruikers die hetzelfde wachtwoord gebruiken voor meerdere diensten, kan hier het risico ontstaan dat een aanvaller toegang krijgt tot accounts door middel van “credential stuffing”.

Zelfs het kraken van hashes van oude wachtwoorden kan tot risico’s leiden. Wanneer een gebruiker wordt gevraagd zijn/haar wachtwoord te veranderen, komt het vaak voor dat het nieuwe wachtwoord een variatie is op het oude wachtwoord. Met oude gekraakte wachtwoordhashes is het soms dus ook mogelijk om het nieuwe wachtwoord te voorspellen. Dit leidt ertoe dat het een goede optie is om bij een kraaktest ook permutaties op oude gekraakte wachtwoorden te testen. Zo kunnen deze voorspelbare patronen bij het veranderen van een wachtwoord mogelijk ook worden gedetecteerd.

Wat zijn de beperkingen van “hash cracking”?

De effectiviteit van “hash cracking” neemt sterk afhankelijk van de complexiteit van een wachtwoord. Hierbij speelt vooral de lengte en complexiteit van een wachtwoord een grote rol. Ook kan een sterk hashingalgoritme, zoals *bcrypt* of *argon2*, de effectiviteit van “hash cracking” sterk verminderen. Dit soort hashingalgoritmes zorgen ervoor dat er relatief veel rekenkracht en werkgeheugen nodig is om een hash te berekenen. Hierdoor is er veel tijd nodig om een aanval succesvol uit te voeren. Ook kunnen hashes extra worden beveiligd door middel van “salting” en “peppering”. Dit voorkomt dat twee dezelfde wachtwoorden ook dezelfde hash opleveren.

Het invoeren van deze beperkingen zijn vooral relevant wanneer er bij een datalek wachtwoordhashes worden buitgemaakt. Wanneer wachtwoordhashes niet op een geschikte manier worden opgeslagen, is de kans groter dat deze op grote schaal kunnen worden gekraakt. Deze gekraakte wachtwoorden kunnen vervolgens weer worden gebruikt in aanvallen met “credential stuffing”.

Welke verschillende methoden zijn er voor “hash cracking”?

Hashes kunnen worden gekraakt volgens de zogenaamde “brute force” methode, waarbij er zo veel mogelijk combinaties van karakters worden gehasht om te kijken of er een overeenkomst wordt gevonden. Een andere methode is om wachtwoordkandidaten te genereren met behulp van woordenlijsten en wachtwoordregels. Het voordeel van deze methode is dat hiermee veel doelgerichter wordt gezocht op wachtwoorden waardoor er in minder tijd soms meerdere wachtwoordhashes kunnen worden gekraakt ten opzichte van de “brute force” methode. Wachtwoordregels maken het mogelijk om ook menselijke patronen toe te passen op woordenlijsten. Denk aan een woord waarbij op het einde een aantal cijfers zijn toegevoegd of een variatie van hoofdletters en kleine letters. Ook kan er gebruik worden gemaakt van “rainbow tables”. Dit zijn tabellen waarbij een groot deel van het rekenwerk al is opgeslagen. Bij het gebruik van “rainbow tables” is er dus minder rekenkracht nodig, maar wel veel opslag om de resultaten van de vele berekeningen op te slaan.

Wat zijn de verschillen tussen “hash cracking” methodes op het gebied van snelheid en efficiëntie?

In veel gevallen heeft “hash cracking” een groot snelheidsvoordeel wanneer er gebruik wordt gemaakt van een moderne GPU. De “brute force” methode kan hierbij in veel gevallen de meeste hashes in een bepaalde tijd doorlopen. Deze methode is echter niet erg doelgericht en is vaak dus alleen effectief bij kortere (<8 karakters) wachtwoorden. Het gebruik van woordenlijsten met wachtwoordregels is vaak minder snel, maar wel doelgerichter. Hierdoor is dit een efficiëntere methode voor het aanvallen van langere (>8 karakters) wachtwoorden. In het geval van NTLM hashes lijken “Rainbow Tables” lijken vooral het efficiëntst voor het aanvallen van wachtwoorden van 8 karakters.

Er zijn ook snelheids- en efficiëntieverschillen die afhankelijk zijn van de gebruikte software. Hashcat leek het meest efficiënte programma te zijn voor het kraken van hashes op de “traditionele” manier. Bij het uitvoeren van een “brute force” aanval van 8 karakters lijkt het gebruik van “rainbow tables” echter het efficiëntst. Wachtwoorden van 1 tot en met 7 karakters kan dit nog prima met hashcat worden gedaan. Wachtwoorden van meer dan 8 karakters kunnen niet in een realistische tijd worden getest met de “brute force” methode. Hiervoor kan hashcat worden gebruikt in combinatie met een goede woordenlijst waarop “rules” kunnen worden toegepast.

Dit geeft de mogelijkheid om een hybrideoplossing te implementeren waarbij verschillende kraakprogramma’s worden ingezet op de “keyspace” waar deze het meest efficiënt zijn. Een voorbeeld hiervan is het gebruik van hashcat voor een brute force op de “keyspace” van 1 tot en met 7 karakters en vervolgens “rainbow tables” te gebruiken voor de keyspace van 8 karakters. Vervolgens kan er opnieuw gebruik worden gemaakt van een woordenlijst met “rules”. Zo kan er een krachtige test worden ingericht waarbij het ook mogelijk is om meerdere aanvallen parallel te laten lopen. Rainbow tables zijn immers vooral afhankelijk van andere hardware dan een kraakprogramma zoals hashcat.

Hoe kan “hash cracking” worden voorkomen om beveiligingsrisico’s tegen te gaan?

“Hash cracking” kan ten eerste worden voorkomen door de infrastructuur waarin wachtwoordhashes worden opgeslagen goed te beveiligen. “Hash cracking” is niet mogelijk wanneer een aanvaller geen toegang heeft

tot de daadwerkelijke wachtwoordhashes. Indien een aanvaller toch toegang kan krijgen tot deze hashes, kan er een poging worden gedaan om deze hashes te kraken. Het succes hiervan kan worden beperkt door in te spelen op de beperkingen van “hash cracking” die in een vorige deelvraag zijn besproken. Denk hierbij aan het gebruik van complexere wachtwoorden, een moeilijker hashalgoritme en het toepassen van “salting” en “peppering”. “Hash cracking” kan ook worden voorkomen door de effectiviteit van de huidige beveiligingsmaatregelen van een organisatie te testen. Dit is bijvoorbeeld mogelijk met een dienst zoals “Password Cracking as a Service”. Zo kunnen mogelijke zwakheden worden gedetecteerd en kan er actie worden ondernomen waar nodig.

Het is echter niet zo dat het voorkomen van “hash cracking” ook leidt tot sterkere wachtwoorden. Het beperkt alleen de succeskans voor deze specifieke aanval. Dit heeft geen effect op methodes zoals “credential stuffing”, die zich ook richten op het aanvallen van zwakke wachtwoorden.

Welke trends zijn er omtrent de bestaande infrastructuur van (potentiële) klanten van HackDefense?

De meeste organisaties maken gebruik van een domain controller voor het beheren van hun gebruikersaccounts. Er zijn verschillende softwareoplossingen die kunnen worden gebruikt voor het opzetten van een domain controller, maar het grootste (>90%) marktaandeel ligt hierbij bij Microsoft Active Directory. Microsoft biedt tegenwoordig ook een versie van Active Directory aan die in de cloud werkt, namelijk Azure Active Directory. Er lijken tegenwoordig steeds meer organisaties te zijn die overstappen naar deze cloudversie. Dit geeft ruimte om in de toekomst te zoeken naar een soortgelijke oplossing die ook toe te passen is voor Azure Active Directory. De besproken methodes voor het uitlezen van wachtwoordhashes zullen hiervoor echter niet werken, aangezien er hierbij minder controle is over een eigen systeem. Er zijn wel mogelijkheden met hybrideoplossingen, waar bij Azure AD wordt gebruikt in combinatie met een domain controller op eigen hardware (Chester, 2019). Dit biedt mogelijkheden voor vervolgonderzoek.

Hoe kan de applicatie wachtwoordhashes verzamelen uit de bestaande infrastructuur van klanten?

Bij Active Directory kunnen wachtwoordhashes worden uitgelezen door een zogenaamde DCSync-aanval uit te voeren of een kopie te maken van de NTDS.dit database door gebruik te maken van de Windows Volume Shadow Copy Service. Voor het project “Password Cracking as a Service” lijkt de DCSync-methode het meest geschikt, omdat er hierbij gebruik kan worden gemaakt van *secretsdump*. Op deze manier is het mogelijk om over een netwerkverbinding informatie vanuit Active Directory te kopiëren, mits er wordt ingelogd met een gebruikersaccount met voldoende rechten.

Hoe kan de applicatie veilig wachtwoordhashes doorsturen naar een (centrale) server?

Om op een veilige manier wachtwoordhashes te versturen, moet er ten eerste alleen worden gecommuniceerd met de centrale server over een versleutelde verbinding. Om deze verbinding te versleutelen kan er gebruik worden gemaakt van https. Op deze manier wordt er niet alleen over een versleutelde verbinding gecommuniceerd, maar kan ook de identiteit van de kraakserver worden geverifieerd. Verder kan de data worden gepseudonimiseerd voordat de hashes naar de kraakserver worden verstuurd. In het product wordt dit gedaan door een uniek id (uuid) te genereren voor elke set aan hashes die naar de kraakserver wordt verstuurd. De kraakserver bewaart vervolgens alleen het uuid en houdt geen informatie bij van de identiteit van de klant.

Hoe kan een organisatie op een veilige manier worden geïnformeerd worden indien er zwakke wachtwoorden zijn gevonden?

Wanneer data wordt geanonimiseerd of gepseudonimiseerd is het voor de kraakserver niet meer mogelijk om uit zichzelf data naar de juiste persoon terug te sturen wanneer een taak is voltooid. De aanvrager van de taak kan echter wel periodiek de status van een taak opvragen met het eerder gegenereerde uuid. Aangezien het hierbij niet mogelijk is voor de kraakserver om de identiteit van de aanvrager te controleren, kan de beveiliging worden verbeterd door niet de volledige zwakke hashes terug te sturen. In plaats hiervan kunnen de hashes worden “afgeknipt”. Dit is een soortgelijke techniek aan “K-anonymity”. Voor de client is het vervolgens mogelijk om de afgeknipte hashes terug te herleiden naar de volledige hashes, maar voor een willekeurig persoon niet.

Hoe kan er automatisch actie worden ondernomen wanneer er een zwak wachtwoord is gevonden?

Wanneer een wachtwoord is gekraakt, kan er automatisch actie op het bijbehorende account worden genomen. Binnen Active Directory is het bijvoorbeeld mogelijk om een wachtwoordreset af te dwingen of een account uit te schakelen. Deze acties kunnen automatisch worden toegepast via het LDAP-protocol, als de cliëntapplicatie over de juiste rechten beschikt. Ook is het mogelijk om via LDAP-informatie op te vragen uit Active Directory, zoals het e-mailadres van een gebruiker of de “Organisational Unit” (OU) waar een gebruiker toe behoort. Deze informatie maakt het mogelijk om automatisch e-mailberichten te versturen om gebruikers te informeren over een zwak wachtwoord of acties kunnen specifiek worden toegespitst op verschillende afdelingen binnen een organisatie op basis van OU.

Hoe kunnen de resultaten van de “kraaktest” op een gebruiksvriendelijke manier worden gepresenteerd aan de klant?

Om bijvoorbeeld ook het management of Security Officers te informeren, kan er automatisch een rapport worden gegenereerd door gebruik te maken van LaTeX in combinatie met Jinja. Deze modules worden in het programma gebruikt om automatisch een kort rapport te genereren in pdf-formaat met de resultaten van kraaktest en een samenvatting van de acties die op bepaalde gebruikersaccounts zijn ondernomen.

Een ander voorbeeld voor het weergeven van resultaten is het weergeven van statistieken in een dashboard. Hierdoor kan er veel informatie worden weergegeven en zo is het vaak mogelijk om extra inzichten te verkrijgen door filters toe te passen. Het nadeel hiervan is echter dat het vaak meer tijd kost om de belangrijkste informatie te vinden en iemand initiatief moet nemen om het dashboard daadwerkelijk te bekijken. Hierom lijkt een kort en beknopt rapport een beterere oplossing. Een rapport geeft meteen antwoord op de vraag of er zwakke wachtwoorden zijn gevonden en of het probleem is opgelost of dat er extra actie moet worden ondernomen. Dit bespaart tijd en moeite en laat meteen zien of er toch nog extra actie moet worden ondernomen. Deze optie is vaak dus sneller en duidelijker en hierdoor is de kans groter dat er daadwerkelijk naar de resultaten wordt gekeken.

Is de applicatie forensisch betrouwbaar?

Tijdens de ontwikkeling van het “proof of concept” zijn er een aantal zaken aangetroffen die in bepaalde gevallen soms onvoorspelbaar reageren. Het programma controleert op het onvoorspelbare gedrag en houdt hier rekening mee. In de overige tests naar forensische correctheid is er geen nieuw onbetrouwbaar of onvoorspelbaar gedrag aangetroffen.

10 Competenties

10.1 A-competenties

10.1.1 Onderzoek

De vraagstelling van het hele onderzoek is opgebouwd uit een enkele hoofdvraag en meedere deelvragen. Deze deelvragen zijn

Voor het uitvoeren van het literatuuronderzoek is er gebruik gemaakt van de sneeuwbalmethode voor het zoeken van extra bronnen. Dit heeft ook geleid tot nieuwe deelvragen die zijn opgenomen in het afstudeerplan en het onderzoeksrapport voor het literatuuronderzoek. Door het toepassen van deze methode is er onderzoek uitgevoerd in meerdere iteraties. Hierbij leid elke iteratie tot nieuwe onderwerpen of extra verdieping. Om het overzicht tijdens dit onderzoek te bewaren en tunnelvisie te voorkomen is het verloop van het onderzoek vastgelegd in een “mindmap”. Deze mindmap geeft ook een visualisatie van de verschillende iteraties van het onderzoek. Het onderzoek is beschreven in de bijlage “*Literatuuronderzoek*”.

In veel documenten wordt er verwezen naar externe bronnen om de inhoudelijke tekst te onderbouwen of te verduidelijken. Aan het eind van deze documenten is er een literatuurlijst aanwezig waarin alle bronnen volgens de APA-norm worden opgesomd.

Om de eisen en wensen voor het product in kaart te brengen zijn er meerdere semi-gestructureerde interviews gehouden met verschillende belanghebbenden bij het product. Hierbij is gebruik gemaakt van open en gesloten vragen die toegespitsts zijn op de verschillende belanghebbenden. Aan de hand van deze interviews is een lijst met functionele en niet-functionele requirements opgesteld die vervolgens zijn geprioriteerd volgens de MoSCoW-methode. De interviews en requirementanalyse zijn beschreven in de bijlage “*Requirementanalyse*”.

10.1.2 Leren leren

Ik heb mijn opdracht grotendeels zelfstandig kunnen uitvoeren. Binnen de afstudeeropdracht was er veel ruimte voor het inbrengen van eigen ideeën en het doen van eigen onderzoek. Dit heeft ervoor gezorgd dat ik weinig extra ondersteuning nodig had. De ondersteuning waar ik om vroeg was vooral voor het verifiëren van ideeën of het feedback vragen over documenten. Hierdoor merkte ik wel dat er soms periodes waren waar ik weinig contact had met anderen. Dit is ook genoemd bij de tussenevaluatie. Ik heb hieraan gewerkt door vaker over mijn ideeën en voortgang te praten met andere medewerkers, ook al had ik geen vragen of problemen.

Verder probeerde ik tijdens mijn opdracht veel afwisseling te zoeken tussen het programmeren van het proof of concept en het schrijven van documentatie. Ik merkte aan mezelf dat ik me beter kon motiveren als ik minder lang met dezelfde taak bezig was. Dit was niet iets waar ik in mijn originele planning

rekening mee heb gehouden, maar ik heb gemerkt dat ik beter werk kon leveren als ik op deze manier werkte.

Tot slot heb ik ook een kennissessie gegeven waarin ik mijn onderzoek en mijn proof of concept presenteerde aan alle werknemers. Hierbij leek iedereen erg positief en geïnteresseerd in mijn uitgevoerde werk en leverde dit ook leuke nieuwe inzichten voor mezelf en voor andere medewerkers.

10.1.3 Professioneel werken

Gedurende het hele afstudeertraject heb ik mij ingezet om me op een professionele wijze op te stellen en methodisch te werken. Bij het opstellen van het afstudeerplan is er een probleemanalyse uitgevoerd om het probleem achter het probleem te beschrijven. Dit is uiteindelijk verder onderzocht en uitgewerkt in het literatuuronderzoek en de gehouden interviews. Hieruit bleek onder andere dat veel mensen gebruiksvriendelijkheid en gemak vaak verkiezen boven beveiliging. Zo wordt het belang van een sterk wachtwoord voor sommige mensen op de achtergrond geplaatst. Hierdoor kan de beveiliging of vertrouwelijkheid van gevoelige gegevens in gevaar komen, wat kan leiden tot financiële schade of schade aan het imago van een organisatie. Dit probleem is gebruikt als drijfveer voor de oplossing, namelijk een dienst die zwakke wachtwoorden kan opsporen en automatisch actie ondernemen voordat er schade ontstaat. Hiernaast kan dit zorgen voor betere bewustwording van de medewerkers zelf, doordat zij een praktijkvoorbeeld zien waaruit de zwakheid van hun wachtwoord blijkt.

Bij het afstudeerplan is er ook een planning opgesteld in de vorm van een Gannt-chart. Gedurende het hele afstudeertraject is er volgens deze planning gewerkt. Hierbij is er elke week een doel gekozen om in die week te behalen. Denk hierbij aan het uitwerken van bepaalde documenten of het ontwikkelen van een specifieke functie van het proof of concept. Aan het eind van de week heb ik op dit doel gereflecteerd en heb ik de planning aangepast op mijn voortgang. Zo kon de planning iedere week worden bijgesteld als dit nodig was. In mijn planning heb ik ook rekening gehouden met het feit dat het bijna niet mogelijk is om het hele afstudeertraject precies in te plannen en dat er een aantal activiteiten moeten worden verschoven. Hierdoor heb ik nooit het gevoel gehad dat ik in tijdnood kwam. Wel heb ik regelmatig de planning voor mezelf aangepast omdat ik merkte dat meer afwisseling een positief effect had op mijn motivatie. Als resultaat hiervan kwam het vaak voor dat ik aan meerdere dingen tegelijk werkte zonder een ander onderdeel eerst af te ronden. Door het stellen van wekelijkse doelen en hier steeds op te reflecteren ging dit echter prima zonder dat mijn planning chaotisch werd.

De bovenstaande methode is gebruikt om een algehele structuur aan te brengen in het afstudeertraject. Verder is de opdracht in het plan onderverdeeld in meerdere fases. Voor elke fase is hierbij een aparte methode geselecteerd. Zo is dit ook beschreven bij de terugkoppeling op de verschillende B-competenties en het hoofdstuk Aanpak. Wel merkte ik dat ik in sommige gevallen niet de meest geschikte methode had gekozen. Dit was vooral duidelijk bij het ontwikkelen van het proof of concept waarbij ik origineel voor de watervalmethode had gekozen. Hierbij ontstond het probleem dat het moeilijk was om een volledig plan te maken voor alle functies, aangezien het proof of concept deels werd gebruikt om de beste methode voor sommige functies te testen. Uiteindelijk heb ik ervoor gekozen om de watervalmethode toch niet te gebruiken, maar in plaats hiervan te wisselen naar “agile”. Dit was voor mij een geschiktere methode, omdat hierbij steeds aan een enkel onderdeel kon worden gewerkt.

Dit werkte beter voor het ontwikkelen, testen en documenteren van de verschillende onderdelen van het proof of concept en heeft ervoor gezorgd dat ik ook veel optionele onderdelen kon onderzoeken en uitwerken.

De communicatie tussen mijzelf en de afstudeerorganisatie is soepel verlopen. Aan het begin van elke werkdag werd er een “stand-up” gehouden waarbij er elke dag werd besproken wat er op die dag werd gedaan. Hier ben ik zelf ook altijd aanwezig geweest en kon ik bespreken wat mijn voortgang was en wat mijn volgende stappen waren. Dit was ook een goed moment om meer te leren over de werkzaamheden van alle medewerkers binnen de organisatie. Verder zijn kantoordagen ingepland met samen met de begeleiders, zodat het altijd makkelijk was om een begeleider aan te spreken wanneer dit nodig was. Overige communicatie verliep telefonisch of via de berichtendienst “Signal”. Hierbij is er gebruik gemaakt van verschillende groepen, onder andere voor medewerkers in het algemeen en een groep voor mezelf en mijn stagebegeleiders. Zo is er ook zelfstandig contact opgenomen met de verschillende stakeholders voor interviews en het uitwisselen van andere ideeën. Ik heb tijdens het uitvoeren van mijn opdracht wel gemerkt dat ik soms weinig contact had met andere stagiairs. Hier heb ik uiteindelijk aan gewerkt door meer interesse te tonen in stageopdrachten van anderen en ook meer over mijn eigen opdracht te delen met medestagiairs. Deze zaken zijn onder andere terug te vinden in de bijlagen “*Tussenevaluatieformulier*” en “*Eindevaluatieformulier*”.

Qua houding heb ik ook altijd mijn best gedaan om me professioneel op te stellen. Zo heb ik regelmatig om feedback gevraagd over mijn werkwijze en mijn geleverde producten. Ik probeer hierbij kritisch te zijn over mijn eigen werk, met het doel hier verbetering in aan te kunnen brengen. Hierbij heb ik ook feedback en waardering geprobeerd te geven tegenover de organisatie zelf waar dit toepasselijk was. De afstudeerorganisatie heeft mij de opdracht vanaf het begin bijna volledig zelfsturend uit laten voeren. Dit is ook terug te lezen in de bijlagen “*Tussenevaluatieformulier*” en “*Eindevaluatieformulier*”.

Tot slot heb ik ook meerdere malen gereflecteerd op mijn eigen werkzaamheden en producten. Dit was soms volledig zelfstandig, maar ook samen met medewerkers en de stagebegeleiders tijdens stagegesprekken. Hierdoor heb ik meerdere malen kunnen werken aan verbeteren van werkwijze, gebruikte methodes en communicatie met anderen. Verdere reflectie is opgenomen in de overige kopjes over competenties en in hoofdstuk 11 Reflectie. Een aantal van de verbeteringen zijn ook terug te lezen in de bijlagen “*Tussenevaluatieformulier*” en “*Eindevaluatieformulier*”.

10.1.4 Innovatie

Tijdens de afstudeeropdracht is er gewerkt aan een volledig nieuwe dienst. Hierdoor bestaat het proof of concept bijna volledig uit eigen ideeën voor de implementatie van de verschillende functies. Hierin heb ik een onder andere een zelfbedachte techniek gebruikt voor het anonimiseren van de resultaten van een kraaktest.

Verder heb ik ook mijn best gedaan mijn nieuwe ideeën te delen met de werknemers van de afstudeerorganisatie en heb ik bij het presenteren van mijn product ook om feedback en ideeën van anderen gevraagd.

Tot slot heb ik ook een aantal risico's genomen door me te richten op de ontwikkeling van de dienst in het algemeen. Oorspronkelijk zou ik meewerken aan een enkel onderdeel van de dienst, maar door veranderingen in de planning bij de afstudeerorganisatie heb ik ervoor gekozen om een concept te bouwen voor het volledige product. Hierdoor heb ik een risico kunnen vermijden waardoor mijn afstudeeropdracht niet zou worden gehinderd door externe factoren, maar ik wel een grotere taak op mezelf nam. Uiteindelijk is het gelukt om alle belangrijke onderdelen van het product uit te werken en ook een groot deel van de extra functies mee te ontwikkelen.

Innovatie en creativiteit is voornamelijk terug te vinden in het "proof of concept" en de bijbehorende documenten.

10.2 B-Competenties

10.2.1 Infrastructuur analyseren

Voor "Password Cracking as a Service" is er aan het begin van de stage gekeken naar verschillende mogelijkheden voor infrastructuur. Hierbij is er eerst vooral onderzoek gedaan naar trends in de infrastructuur van de klanten, opslag van (gehashte) wachtwoorden, technieken voor het kraken van wachtwoorden en de juridische aspecten van de afstudeeropdracht. Hiernaast is er ook onderzoek uitgevoerd naar verschillende aspecten van informatiebeveiliging.

Hiernaast zijn er interviews gehouden met verschillende stakeholders van het project en er zijn vergaderingen bijgewoond met een potentiële klant van de dienst. Aan de hand van deze gesprekken is er een requirementanalyse uitgevoerd waarbij er verschillende functionaliteiten en eisen voor het project in kaart zijn gebracht. Hierbij zijn de verschillende eisen van verschillende stakeholders geprioriteerd volgens de MoSCoW-methode.

Uiteindelijk is de infrastructuur geselecteerd op basis van terugkoppeling van de requirementanalyse met de organisatie en de uitgevoerde snelheidstests uit hoofdstuk 7.5.3.1 BENCHMARKS. Hierbij is er ook een concept ontwikkeld voor het anoniem communiceren tussen de verschillende systemen binnen de infrastructuur en is er onderzoek gedaan naar de bijbehorende configuratie van Active Directory op het gebied van communicatieprotocollen en toegangsrechten. Zo is er gekozen voor een infrastructuur die in het onderzoek het meest efficiëntst bleek te zijn en er zijn maatregelen geïmplementeerd om beveiligingsrisico's te vermijden.

Het onderzoek doen naar trends en het uitvoeren van een requirementanalyse met de functionele eisen, niet-functionele eisen en behoeftes van verschillende stakeholders is kenmerkend voor deze competentie op niveau 3.

10.2.2 Software analyseren

Een belangrijk onderdeel van de dienst "Password Cracking as a Service" is de software waarmee de wachtwoordhashes daadwerkelijk mee worden gekraakt. Tijdens de afstudeeropdracht zijn er verschillende kraaktools vergeleken om de verschillen tussen deze programma's in kaart te brengen.

Eerst is er gekeken naar de bestaande onderdelen binnen de afstudeerorganisatie die worden gebruikt voor het kraken van wachtwoordhashes. Dit is gedaan door het houden van interviews binnen de organisatie en het bijwonen van vergaderingen met een potentiële klant. Hierbij is er inzicht verkregen in de bestaande aanpak rondom het kraken van wachtwoordhashes en is er meer inzicht verkregen in de wijze waarop de verschillende belanghebbenden baat hebben bij de dienst “Password Cracking as a Service”.

Om een goede analyse uit te kunnen voeren is er een vergelijking gemaakt tussen verschillende kwaliteitsnormeringen voor softwarekwaliteit. Op basis van deze vergelijking is de meest moderne en toepasselijke ISO-norm geselecteerd.

De verschillende kenmerken uit de kwaliteitsnormering zijn meegenomen in een requirementanalyse. Hierbij is er rekening gehouden met de inbreng van verschillende stakeholders afkomstig uit de interviews en vergaderingen. Aan de hand van deze gesprekken zijn de verschillende kwaliteitskenmerken geprioriteerd en deze zijn gebruikt om de verschillende softwaretests op te stellen. Aan de hand van dezelfde kwaliteitskenmerken zijn ook acceptatiecriteria opgesteld. Hierbij is ook een risicoanalyse uitgevoerd om van vooraf maatregelen te kunnen nemen om problemen te voorkomen waardoor mogelijk niet aan de acceptatiecriteria kon worden voldaan.

Ook is er gekeken naar de forensische correctheid van de software. Zo is er gekeken of de verschillende kraaktools consistent hetzelfde antwoord geven. Hiernaast is er gekeken of de kraaktools goed te gebruiken zijn in een groter systeem door ze te proberen te gebruiken vanuit het “proof of concept”. Dit geeft ook inzicht in mogelijk onvoorspelbaar gedrag dat kan ontstaan door slechte communicatie tussen de kraaktools en andere software. Aangezien de kraaktools in een volledig automatisch proces moet kunnen worden ingezet, is dit een waardevol onderdeel.

De resultaten van de tests zijn ook gerapporteerd in het adviesrapport. De resultaten zijn schematisch vastgelegd in tabellen en diagrammen met daarbij een beschrijving van hoe de gegevens geïnterpreteerd moeten worden. De informatie uit deze tabellen is vervolgens gebruikt om een conclusie te trekken en de verschillende deelvragen te beantwoorden.

Het uitvoeren van een requirementanalyse voor een softwaresysteem met verschillende belanghebbenden en het definiëren van acceptatiecriteria aan de hand van kwaliteitseigenschappen en een risicoanalyse zijn kenmerkend voor deze competentie op niveau 3.

10.2.3 Software adviseren

Resulterend aan de afstudeeropdracht is er uiteindelijk een advies opgesteld voor de ontwikkeling van “Password Cracking as a Service”. Dit advies bestaat uit een adviesrapport samen met het “proof of concept”. Dit “proof of concept” dient als ondersteuning van het advies en geeft een voorbeeldimplementatie van mogelijke onderdelen en functionaliteiten van het product.

Een belangrijk onderdeel van het advies is de selectie van een geschikt kraakprogramma. Hierbij is er niet alleen gekeken naar verschillen tussen programma’s, maar ook verschillende aanpakken en kraaktechnieken die hierbij kunnen worden gebruikt. Dit heeft uiteindelijk geresulteerd in een advies

voor een aanpak waarbij verschillende kraaktools kunnen worden gecombineerd, zodat de voordelen van verschillende technieken kunnen worden toegepast.

Een andere belangrijke factor voor het selecteren van software is het volledige kostenplaatje. Dit gaat verder dan alleen de aanschaf- en opzetkosten van de software. Voor deze analyse is er ook gekeken naar extra kosten die nodig zijn op het gebied van hardware. Hiernaast is er ook gekeken naar energieverbruik en efficiëntie van verschillende kraaktools. Zo is er een schatting gemaakt van de energiekosten voor het uitvoeren van een kraaktest. Hierbij is er ook gekeken naar de impact van het energieverbruik op het milieu en is dit meegenomen in het advies. Zo kunnen tests worden uitgesteld totdat ze (zo veel mogelijk) met groene stroom kunnen worden uitgevoerd.

Het “proof of concept” is uiteindelijk getest tegen de acceptatiecriteria. Hierbij zijn de ontwikkelde onderdelen vergeleken met de requirementanalyse en zijn alle functionaliteiten getest aan de hand van een opgesteld testplan. Hierbij lag de nadruk ook op een forensisch correcte werking van het product. Zo zijn er bijvoorbeeld bepaalde onderdelen in Active Directory beschreven waarbij een opdracht niet correct wordt uitgevoerd zonder dat Active Directory hierbij een foutmelding geeft. In het adviesrapport en het “proof of concept” wordt dit gedrag beschreven samen met manieren om dit soort situaties goed af te handelen.

Alle verschillende adviezen en de onderbouwingen hiervan zijn beschreven in het adviesrapport. Het adviesrapport is als bijlage meegeleverd bij dit document.

De uiteindelijke resultaten en adviezen zijn uiteindelijk ook gepresenteerd aan de organisatie met meerdere stakeholders. Hierbij is de motivatie achter de keuzes gepresenteerd en er zijn ideeën uitgewisseld over mogelijke alternatieve aanpakken voor bepaalde onderdelen van de dienst. Ook is er een demo gegeven van het “proof of concept”.

Het adviseren van met betrekking tot de keuze van software architectuur, rekening houdend met kwaliteitskenmerken zoals kosten, prestaties en beveiliging is kenmerkend voor deze competentie op niveau 3. Hiernaast ook het adviseren over een softwareontwikkelproces en het verwerken van een grote hoeveelheid data met aandacht voor privacy.

11 Reflectie

In dit hoofdstuk wordt er gereflecteerd op het volledige afstudeertraject. Hierbij wordt er ten eerste gereflecteerd op de uitgevoerde werkzaamheden met betrekking tot de B-competenties. Hiernaast wordt er ook algemener gereflecteerd op de verschillende onderdelen van de afstudeeropdracht.

11.1 Proces

Aan het begin van het afstudeertraject heb ik planning opgesteld voor het doorlopen van de verschillende fases van de afstudeeropdracht. Bij het opstellen van deze planning heb ik ook bewust extra tijd overgelaten voor mogelijke vertragingen of onverwachte veranderingen tijdens het project. In de planning zorgde dit uiteindelijk voor een aantal situaties waarbij er meerdere weken achter elkaar stonden ingepland voor het schrijven van rapporten en documentatie. In principe stond hier voldoende tijd voor ingepland, maar ik merk dat dit voor mij soms een gebrek aan afwisseling opleverde. Dit kon ten koste gaan van mijn eigen motivatie en daardoor ook mijn eigen productiviteit. Ik heb voor mezelf voor meer afwisseling geprobeerd te zorgen door al eerder te beginnen aan het ontwikkelen van het “proof of concept”. Dit heeft geholpen, maar achteraf gezien had ik in mijn originele planning ook moeten denken aan de invloed van bepaalde keuzes op mijn motivatie en productiviteit. Zo zou ik in de toekomst efficiënter om kunnen gaan met de tijd die voor een opdracht ingepland staat.

In het begin van het afstudeertraject ontstond er ook een plan voor het meewerken aan een onderdeel voor de daadwerkelijke software voor de dienst. Het was hierbij de bedoeling dat deze software ook uiteindelijk aan klanten zou worden geleverd. Na het opstellen van de het afstudeerplan heb ik een aantal vergaderingen bijgewoond met een potentiële afnemer van de dienst. Deze organisatie is ook een partner bij het ontwikkelproces. Uit deze vergaderingen bleek echter dat deze partner meer eisen had dan origineel binnen het afstudeerplan was besproken. Hierdoor zou mijn opdracht veranderen in de fase waarbij ik mee zou werken aan de ontwikkeling van het echte product en zo zou mijn opdracht niet meer goed passen binnen mijn uiteindelijke onderzoek. Uiteindelijk heb ik ervoor gekozen om me meer los te trekken van het meedoen in de ontwikkeling van het product en in plaats hiervan mijn eigen “proof of concept” te maken. Dit heeft ervoor gezorgd dat er veel minder risico is dat de scope van mijn afstudeeropdracht te veel zou veranderen. Dit heeft er alleen wel voor gezorgd dat ik hierdoor minder betrokken was bij de ontwikkeling van het echte eindproduct en mijn afstudeeropdracht meer als advies diende. Achteraf gezien had ik wel meer betrokken geweest willen zijn bij de ontwikkeling van de software. Dit had ik kunnen doen door vaker ideeën uit te wisselen met softwareontwikkelaars van de organisatie of door vaker mijn ideeën over mijn eigen “proof of concept” te delen.

Verder ben ik bij het ontwikkelen van mijn proof of concept ook gewisseld van toegepaste methode. Hierbij heb ik geleerd dat het soms juist averechts werkt om een plan van aanpak streng te volgen. Zo had ik origineel het plan om volgens de waterval-methode te werken, maar ik merkte al gauw dat dit totaal niet geschikt was voor mijn opdracht. Door de verschillende losstaande onderdelen van mijn

proof of concept leek de agile-methode veel meer geschikt, waardoor ik van mijn originele plan ben afgeweken en dit heb moeten aanpassen. Ik merk hierbij dat het goed kan zijn om kritisch te zijn op mijn eigen plannen en dat het verstandig is om andere methodes niet uit te sluiten, ondanks dat deze geen deel uitmaken van het originele plan van aanpak.

11.2 Bedrijfsbegeleiding

Ik vind dat de communicatie met de bedrijfsbegeleiding prima is verlopen. Ik merkte aan het begin van de afstudeerperiode dat ik wel eerst tijd nodig had om te wennen aan de cultuur van de organisatie. In mijn vorige stage heb ik bij een veel grotere organisatie gewerkt waar er veel meer afstand is tussen verschillende afdelingen. Dit zorgde er soms voor dat het moeilijker was om iemand te spreken voor begeleiding. Dit was bij de afstudeerorganisatie totaal niet het geval en daardoor was dit voor mij een groot contrast met mijn vorige stage. Dit heeft ervoor gezorgd dat ik vaak compleet zelfstandig probeerde te werken. Dit lukte prima, maar zorgde er soms wel voor dat ik niet veel communiceerde met begeleiders of medestagiairs. Hierdoor kunnen er situaties ontstaan waardoor mijn begeleiders soms niet goed op de hoogte waren van mijn huidige werkzaamheden. Ik heb hieraan gewerkt door vaker te proberen te delen waar ik mee bezig ben en ook meer interesse te tonen in stageopdrachten van anderen. Achteraf gezien had ik dit meer kunnen verbeteren door ook meer mijn eigen ervaringen en werkhouding te bespreken met mijn begeleiders, naast de besprekingen over de voortgang van mijn opdracht. Dit maakt het voor begeleiders duidelijker waar ik mee bezig ben en kan ook leiden tot meer gerichte begeleiding.

In het algemeen ben ik erg positief over de begeleiding vanuit het bedrijf zelf. De organisatie gaf veel ruimte om eigen ideeën te delen en hierbij werd ook veel interesse getoond. Dit was ook het geval bij het presenteren van mijn resultaten en de demo van het “proof of concept”.

11.3 Product

In het algemeen ben ik tevreden met de gemaakte producten. Ik heb hierbij vooral veel moeite gestoken in het ontwikkelen van het “proof of concept”. Ik heb hier meer functies kunnen ontwikkelen dan ik origineel had verwacht. Het ontwikkelen van een aantal onderdelen heeft voor mij ook een hoop informatie gegeven voor de rest van mijn onderzoek. Bijvoorbeeld over de werking van sommige functies in Active Directory of de beperkingen van bepaalde kraakprogramma’s.

Ik heb wel gemerkt dat ik het soms moeilijk vond om een deel van de onderdelen van het “proof of concept” ook te beschrijven in mijn verslagen. Dit kwam denk ik vooral doordat er veel technische overwegingen zijn geweest die moeilijk te begrijpen zijn zonder de achterliggende code ook te begrijpen. Ik heb dit geprobeerd op te lossen door dit in mijn verslagen op te delen per functionaliteit en vooral de belangrijkste bevindingen per onderdeel uit te werken. Dit leek mij de beste oplossing, alhoewel dit de samenwerking tussen de verschillende onderdelen minder goed laat zien. Achteraf gezien had ik denk ik vaker mijn werk willen delen. Vooral om beter te kunnen delen hoe ik mijn producten heb ontwikkeld en duidelijk te maken welke uitdagingen hierbij hoorden.

11.4 Competenties

Achteraf gezien denk ik dat het bewijzen van mijn competenties prima is verlopen. Mijn afstudeeropdracht bestond uit verschillende fases die elk redelijk goed binnen mijn gekozen B-competenties leken te vallen. Ik had soms wel problemen om de rubrics van sommige B-competenties goed te begrijpen. Dit kwam denk ik vooral omdat er in deze rubrics niet erg gericht zijn op een volledig nieuw product, maar meer de verbetering van een onderdeel van een bestaand product or proces. Ook het uitvoeren van een kostenanalyse (TCO) vond ik niet makkelijk verlopen, aangezien ik tijdens mijn opdracht alleen maar gebruik heb gemaakt van gratis open source programma's in combinatie met een zelfgemaakt programma. Hierom heb ik uiteindelijk ook gekeken naar de energiekosten en eigen schaal voor toepasbaarheid per kraakprogramma.

Het aantonen van A-competenties ging naar mijn idee ook prima. Ik vond het moeilijker om dit goed te beschrijven, omdat zaken zoals "professioneel werken" moeilijk zijn toe te wijzen aan een specifieke fase van mijn opdracht. Ook heb ik het idee dat zaken zoals zelfstandigheid juist voor uitdagingen kunnen zorgen. Aangezien ik mijn opdracht grotendeels zelfstandig heb uitgevoerd, moest ik opletten dat ik andere mensen genoeg bij mijn opdracht betrokken hield.

11.5 Onderwijs

Ik vind dat de communicatie met het onderwijs tijdens de afstudeerperiode prima is verlopen. Ik vind dat er weinig contactmomenten zijn geweest met de begeleiding van de hogeschool, maar ik heb hier zelf geen problemen van ondervonden. Ik heb wel het idee dat het beter zou zijn geweest als ik vaker mijn voorgang deelde met mijn stagebegeleider. Ik had zelf het idee dat ik vooral contact op wilde nemen wanneer er problemen ontstonden of wanneer ik zelf vragen had. Deze situaties kwamen echter niet veel voor, waardoor ik zelf ook weinig contact zocht. Dit heeft voor mijn opdracht niet voor specifieke problemen gezorgd, maar ik denk dat het beter zou zijn geweest als ik wel vaker contact had gezocht om de voortgang van mijn opdracht door te geven. Ondanks dat ik zelf het idee heb dat ik geen hulp nodig heb, kan feedback van anderen soms toch waardevolle informatie opleveren. Ik heb in de latere fases van de afstudeeropdracht wel wat vaker contact gezocht met mijn begeleider om voortgang door te geven, maar achteraf gezien denk ik dat het beter zou zijn geweest als ik dit vaker had gedaan. Dit is een soortgelijke situatie als de reflectie over de bedrijfsbegeleiding. Door vaker mijn voortgang te delen met anderen kan ik meer feedback ontvangen voor mijn werk. Hiermee is het voor mij mogelijk om in de toekomst beter werk te leveren en mijn werk beter af te stemmen met anderen.

Verder leek de intervisie mij een goed idee, ondanks dat ik zelf niet veel specifieke vragen had waar ik het met andere studenten over zou willen hebben. Het lijkt me vooral waardevol als er bij de intervisie een aantal (oude) verslagen centraal worden behandeld. Hierbij kunnen bijvoorbeeld goede en slechte voorbeelden worden gegeven. In dit geval ontstaan er bij andere studenten denk ik ook meer vragen over hun eigen verslagen en wordt het makkelijker om problemen te identificeren en aan te pakken.

12 Nawoord

Erik heeft laten zien dat hij individueel een professionele stageopdracht goed kan uitvoeren. Hij heeft ons weten overtuigen de opdracht op zijn manier uit te voeren en had daarvoor amper begeleiding voor nodig.

Daarnaast past Erik goed binnen het team door zijn zelfredzaamheid en de hoeveelheid kennis die hij in korte tijd heeft opgebouwd. Dit heeft ook geresulteerd in een functie binnen HackDefense.

Wij zijn beide erg benieuwd hoe Erik zich verder gaat ontwikkelen m.b.t programmeren en hacken. Ook zal Erik zijn stageproduct verder ontwikkelen waardoor wij het product beter op de markt kunnen brengen. Wij denken dat Erik deze verantwoordelijkheid goed zal oppakken met de expertise die hij heeft opgebouwd tijdens zijn stagetraject.

Sander Meijering en Niels Eris – 13 juni

13 Begrippenlijst

Dit hoofdstuk geeft een omschrijving van een aantal begrippen die in dit document zijn gebruikt.

- **Domain Controller**: een serversysteem dat wordt ingezet voor het centraal beheren van onder andere gebruikersaccounts binnen een domein.
- **Active Directory**: een verzameling aan software van Microsoft die kan worden ingezet als Domain Controller.
- **Hashingalgoritme:** een wiskundig onomkeerbaar algoritme. Kan onder andere worden gebruikt voor het veilig opslaan van wachtwoorden.
- **Hash(waarde)**: de uitkomst van een hashingalgoritme.
- **NTLM**: “NT LAN Manager”. Een methode voor authenticatie binnen Active Directory. NTLM verwijst ook naar de hashwaarde die bij deze methode van authenticatie wordt gebruikt.
- **Principle of Least Privilege**: een principe uit de informatiebeveiliging die beschrijft hoe een onderwerp alleen de minimale bevoegdheden mag krijgen die hij/zij nodig heeft.
- **Security Through Obscurity:** een concept dat gegevens alleen worden beveiligd door ze te verbergen voor het publieke oog.
- **Hashrate:** snelheid waarmee hashes kunnen worden berekend.

14 Verwijzingen

Agconnect. (2020, November 2). *Gemak verslaat beveiliging, ook bij de overheid*. Opgehaald van Agconnect: https://www.agconnect.nl/artikel/thuiswerk-ict-gemak-verslaat-beveiliging-ook-bij-de-overheid

Agile Scrum Group. (sd). *Minimum Viable Product en Scrum*. Opgehaald van Agile Scrum Group: https://agilescrumgroup.nl/minimum-viable-product/

Artikel 10: Privacy. (2018). Opgehaald van De Nederlandse Grondwet: https://www.denederlandsegrondwet.nl/id/vkugbqvdtdwz/artikel_10_privacy

Autoriteit persoonsgegevens. (2021, november 12). *AP beboet Transavia om slechte beveiliging persoonsgegevens*. Opgehaald van Autoriteit persoonsgegevens: https://autoriteitpersoonsgegevens.nl/nl/nieuws/ap-beboet-transavia-om-slechte-beveiliging-persoonsgegevens

Autoriteit persoonsgegevens. (sd). *Verantwoordingsplicht*. Opgehaald van Autoriteit persoonsgegevens: https://autoriteitpersoonsgegevens.nl/nl/onderwerpen/algemene-informatie-avg/verantwoordingsplicht

Autoriteit persoonsgegevens. (sd). *Verantwoordingsplicht*. Opgehaald van Autoriteit persoonsgegevens: https://autoriteitpersoonsgegevens.nl/nl/onderwerpen/algemene-informatie-avg/verantwoordingsplicht

Autoriteit Persoonsgegevens. (sd). *Wat zijn persoonsgegevens?* Opgehaald van Autoriteit Persoonsgegevens: https://autoriteitpersoonsgegevens.nl/nl/over-privacy/persoonsgegevens/wat-zijn-persoonsgegevens

Broekhoven, M. v. (sd). *Pseudonimiseren of anonimiseren?* Opgehaald van Juridict: https://www.juridict.nl/juridict-nieuwsartikel/pseudonimiseren-of-anonimiseren/

Carnegie Mellon University. (2020). *Practical Recommendations for Stronger, More Usable.* Opgehaald van Carnegie Mellon University: https://www.andrew.cmu.edu/user/nicolasc/publications/Tan-CCS20.pdf

Chester, A. (2019, Februari 18). *Azure AD Connect for Red Teamers*. Opgehaald van XPNSec: https://blog.xpnsec.com/azuread-connect-for-redteam/

CISA. (2013, Mei 10). *Least Privilege*. Opgehaald van Cybersecurity & Infrastructure Security Agency: https://www.cisa.gov/uscert/bsi/articles/knowledge/principles/least-privilege

Engelfriet, A. (2018, November 6). *De Wet Computercriminaliteit: Computervredebreuk*. Opgehaald van Ius Mentis: https://www.iusmentis.com/beveiliging/hacken/computercriminaliteit/computervredebreuk/

Gerechtshof Amsterdam. (2005, Maart 8). *ECLI:NL:GHAMS:2005:AS9143*. Opgehaald van Rechtspraak: https://uitspraken.rechtspraak.nl/inziendocument?id=ECLI:NL:GHAMS:2005:AS9143

Gerechtshof Arnhem-Leeuwarden. (18, Mei 2015). *ECLI:NL:GHARL:2015:2954*. Opgehaald van Rechtspraak: https://uitspraken.rechtspraak.nl/inziendocument?id=ECLI:NL:GHARL:2015:2954

ICT Recht. (2013, September 13). *Is een (gehasht) wachtwoord een persoonsgegeven?* Opgehaald van ICT Recht: https://www.ictrecht.nl/blog/is-een-gehasht-wachtwoord-een-persoonsgegeven

Kader. (sd). *ISO 27001 en de relatie met AVG wetgeving*. Opgehaald van Kader Advies: https://kader-advies.nl/informatiebeveiliging/iso-27001-certificering/iso-27001-en-avg-wetgeving

Maxius. (2016, April 20). *Artikel 350a Wetboek van Strafrecht*. Opgehaald van Maxius: https://maxius.nl/wetboek-van-strafrecht/artikel350a

Maxius. (2016, April 20). *Artikel 350c Wetboek van Strafrecht*. Opgehaald van Maxius: https://maxius.nl/wetboek-van-strafrecht/artikel350c

Microsoft. (sd). *Client, service, and program issues can occur if you change security settings and user rights assignments*. Opgehaald van Microsoft Support: http://go.microsoft.com/fwlink/?LinkId=17936

NIST. (2008, Juli). *Guide to General Server - Recommendations of the National Institute of Standards and Technology*. Opgehaald van NIST: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-123.pdf

Overheid.nl. (sd). *Verdrag tot bescherming van de rechten van de mens en de fundamentele vrijheden*. Opgehaald van Overheid.nl: https://wetten.overheid.nl/BWBV0001000/2010-06-10/0/Verdrag_2/Verdragtekst/TiteldeelI/Artikel8/afdrukken

Overstappen.nl. (2022, Mei). *Stroomprijs*. Opgehaald van Overstappen.nl: https://www.overstappen.nl/energie/stroomprijs

PrivazyPlan. (sd). *Artikel 4 EU-AVG "Definities"*. Opgehaald van PrivazyPlan: https://www.privacy-regulation.eu/nl/artikel-4-definities-EU-AVG.htm

Rijksoverheid. (2021, juli 14). *Onderzoek wijst uit: thuiswerken is een blijvertje*. Opgehaald van Rijksoverheid: https://www.rijksoverheid.nl/actueel/nieuws/2021/07/14/onderzoek-wijst-uit-thuiswerken-is-een-blijvertje

15 Bijlagen

In dit hoofdstuk wordt een opsomming gegeven van de bijlagen die bij dit document worden meegeleverd.

1. Afstudeerplan.pdf
2. Literatuuronderzoek.pdf
3. Requirementsanalyse.pdf
4. Testplan.pdf
5. Adviesrapportage.pdf
6. Hashcat benchmark.xlsx
7. Stroomverbruik.xlsx
8. Proof of concept.zip
9. Testplan PoC.pdf
10. Testcases PoC.xlsx
11. Tussenevaluatieformulier.pdf
12. Eindevaluatieformulier.pdf

HackDefense

IT security, maar dan begrijpelijk

HackDefense B.V.
Postbus 3025
2301 DA Leiden
Tel. (071) 204 0101

info@hackdefense.nl
https://hackdefense.nl