Hoe kan trust management effectief worden geïmplementeerd voor het beveiligen van bedrijven met heterogene gesegmenteerde netwerken?

Thibo Haezaert.

Scriptie voorgedragen tot het bekomen van de graad van
Professionele bachelor in de toegepaste informatica

Promotor: Dhr. G. Blondeel
Co-promotor: Dhr. D. Mussen
Academiejaar: 2024–2025
Eerste examenperiode

Departement IT en Digitale Innovatie .

HO
GENT

Woord vooraf

In de voorbije 3 jaren van mijn studies in toegepaste informatica heb ik veel mogen bijleren, dankzij deze kennis mocht mijn passie voor IT nog meer groeien. In mijn laatste jaar kreeg ik de kans om bij KBC mijn stage te lopen, waar ik al gauw zag dat naast mijn opleiding nog veel kon worden bijgeleerd.

Binnen deze stage leerde ik veeltal bij over certificate authorities en public key infrastructures, technologieën die een cruciale rol hebben in de werking van digitale communicatie, maar waarvan ik alleen beschikte over de basiskennis. Mijn co-promotor, Dirk Mussen, wie ook mijn stagementor was, bracht mij dan ook het idee voor het onderzoeksonderwerp van deze bachelorproef. Een onderwerp dat een actueel probleem aanhaalt die vele bedrijven en organisaties treft. Dankzij de kennis opgedaan tijden mijn opleiding, stage en de begeleiding van mijn co-promotor, was het begrijpen en aanpakken van dit probleem een stuk eenvoudiger.

Bij deze wil ik mijn co-promotor graag bedanken voor zijn begeleiding en steun tijdens het uitvoeren van dit onderzoek. Naast mij co-promotor bedank ik ook graag mijn promotor, Gilles Blondeel voor het geven van feedback op deze paper alsook tips voor het mogelijks presenteren van deze bachelorproef. Ik bedankt ook graag alle andere docenten die ik doorheen mijn opleiding heb mogen ontmoeten, voor de kennis die zij met mij hebben gedeeld.

Samenvatting

Het beheren van alle truststores en hun trust (de root certificaten die men vertrouwt), binnen een netwerk kan een tijdrovende taak zijn voor bedrijven. Zeker als het gaat over verschillende end-points met verschillende besturingssystemen, waarbij niet elke end-point dezelfde root certificaten nodig heeft. Het is belangrijk dat bedrijven de root certificaten die ze gebruiken goed beheren, omdat deze certificaten bepalen welke communicatiesystemen vertrouwd worden en welke niet. Het vertrouwen van onnodige certificaten kan leiden tot een verhoogd risico op cyberaanvallen, terwijl het niet vertrouwen van noodzakelijke certificaten kan leiden tot een operationeel risico door het niet kunnen bereiken van bepaalde bronnen.

De vraag die deze bachelorproef beantwoorde is hoe bedrijven de truststores van hun systemen kunnen beheren, waarbij er rekening gehouden wordt met de verschillende besturingssystemen en de verschillende netwerksegmenten binnen een netwerk. Deze bachelorproef streeft ernaar om deze vraag te beantwoorden door het leveren van een oplossingen die bedrijven kunnen gebruiken voor het beheren van de systeem truststores van Windows en Linux end-points. Daarnaast worden er ook aanbevelingen gegeven voor bedrijven die deze oplossingen willen implementeren in hun infrastructuur.

Om tot deze oplossing te komen, werd er eerst een literatuurstudie uitgevoerd die belangrijke concepten en achtergrond informatie aanhaalde en daarnaast ook veel voorbereidende informatie gaf voor het ontwerpen van een oplossing. Na een literatuurstudie werd een proof-of-concept opgezet waarin verschillende tools voor trust management werden getest.
In de proof-of-concept omgeving werden er voor beide Linux als Windows end-points 2 mogelijke oplossingen gevonden. Bij Windows werd er gebruik gemaakt van Group Policy Objects (GPO's) voor de eerste oplossing en System Center Configuration Manager (SCCM) voor de tweede oplossing. Bij Linux werd in de eerste oplossing gewerkt met Ansible en later bij de tweede oplossing met Chef. Beide oplossingen hebben hun eigen voor- en nadelen, maar hun toepasbaarheid hangt af van de infrastructuur van een bedrijf en hun voorkeuren.

De oplossing kan eenvoudig aangepast worden, zodat het bepalen van vertrouwde root certificaten niet beperkt blijft tot netwerksegmentatie, maar ook op basis van andere criteria kan gebeuren. Daarnaast werden er nog een aantal aanbevelingen

gegeven zoals het afstemmen van de truststore updates met activiteiten van een bedrijf hun bestaande PKI's en het verder onderzoeken van de mogelijkheden van de oplossingen op een grotere schaal.

Door de beperkte scope werd niet ingegaan op security-aspecten, IoT-devices of MacOS-systemen. Deze onderwerpen bieden kansen voor verder onderzoek.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Lijst van codefragmenten

1

Inleiding

In moderne bedrijfsomgevingen worden netwerken steeds complexer en diverser. Organisaties bevatten vaak heterogene netwerken, dat wilt zeggen dat binnen het netwerk verschillende soorten apparaten aanwezig zijn die niet allemaal dezelfde besturingssystemen en software gebruiken. Vaak zijn deze netwerken ook opgedeeld in verschillende segmenten op basis van de bedrijfsrollen en -vereisten, om zo de aanvalshoeken te beperken en de veiligheid van het netwerk te verhogen. Een andere preventieve maatregelen die bedrijven vaak nemen is het limiteren van de root certificaten die vertrouwd worden door een systeem, zodanig het enkel de root certificaten vertrouwd die cruciaal zijn voor de werking van het systeem.

Een specifiek probleem binnen deze context is dan hoe men de inhoud van de verschillende truststores over verschillende besturingssystemen kan beheren op een manier waarbij elk systeem enkel de root certificaten vertrouwt die noodzakelijk zijn voor dat systeem zelf. Trust managment wilt dit probleem oplossen door te beheren wie welke root certificaten vertrouwt (welke trust een systeem heeft) en hoe dit kan worden afgedwongen.

1.1. Probleemstelling

Vele bedrijven met grotere diverse netwerken zoals heterogene netwerken en gesegmenteerde netwerken hebben vandaag de dag nogsteeds moeite met het centraal beheren van de (root) certificaten die worden vertrouwd door hun end-points. Dit komt door de verspreide trust stores die gevormt worden door de verschillende gebruikte applicaties en besturingssystemen binnen het netwerk, daarnaast is het ook belangrijk om het aantal vertrouwde certificaten te beperken om de veiligheid van het netwerk te garanderen. Bedrijven hebben dan ook netwerksegmentatie die bepaald is op basis van de rol van de systemen binnen het netwerk, maar ook op

basis van de vertrouwensrelaties die er zijn tussen de systemen. Er wordt gezocht naar een oplossing om de vertrouwde rootcertificaten centraal te beheren, met de mogelijkheid om deze te beperken tot enkel de noodzakelijke vertrouwensrelaties, bijvoorbeeld in het kader van netwerksegmentatie.

1.2. Onderzoeksvraag

Aan de hand van welke combinatie van tools en configuraties kan een trust management-systeem worden opgezet binnen een bedrijf met een heterogeen gesegmenteerd netwerk waarbij de inhoud van de truststores van endpoints centraal kan worden beheerd?
Deelvragen:

- Wat zijn de belangrijkste concepten en technologieën achter PKI en truststorebeheer?
- Wat zijn de verschillende truststores die worden gebruikt door de meest voorkomende besturingssystemen? Hoe kunnen deze worden beheerd?
- Welke tools en technieken kunnen worden gebruikt voor truststorebeheer, en wat zijn hun beperkingen?
- Hoe kan de inhoud van de truststores van endpoints centraal worden beheerd?
- Hoe kan men de vertrouwde certificaten anders beheren voor verschillende netwerksegmenten?
- Welk netwerkplan zal worden gebruikt om de proof-of-concept virtuele omgeving op te zetten?
- Hoe worden de gevonden tools en technieken geïmplementeerd in de proof-of-concept virtuele omgeving?
- Wat is de effectiviteit van de opgezette proof-of-concept virtuele omgeving?
- Welke aanbevelingen kunnen worden gegeven aan bedrijven die een trust management-systeem willen implementeren?

1.3. Onderzoeksdoelstelling

Het doel binnen dit onderzoek is om een proof-of-concept virtuele omgeving op te zetten die systemen met de meest commercieel gebruikte besturingssystemen en netweksegmentatie, waarbij de truststores van de systemen centraal kan worden beheerd en hun inhoud afhankelijk is van het netwerksegment waar ze zich in bevinden. De proof-of-concept omgeving zal als inspiratie dienen voor bedrijven die een trust management systeem willen implementeren, de nadruk binnen dit onderzoek wordt gelegd op de functionaliteit van de oplossing, er zal dus niet te diep

gekeken worden naar de beveiliging van de oplossing. Verdere aanbevelingen voor een implementatie in een productieomgeving zullen ook worden gegeven aan bedrijven.

1.4. Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:
In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.
In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.
In Hoofdstuk 4 wordt een virtuele omgeving opgesteld die aan de hand van de gevonden tools en technieken uit de literatuurstudie een aantal werkende trust management systemen bevat. Aanbevelingen worden meegegeven aan bedrijven voor de implementatie van deze systemen in hun infrastructuur.
In Hoofdstuk 5, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2

Stand van zaken

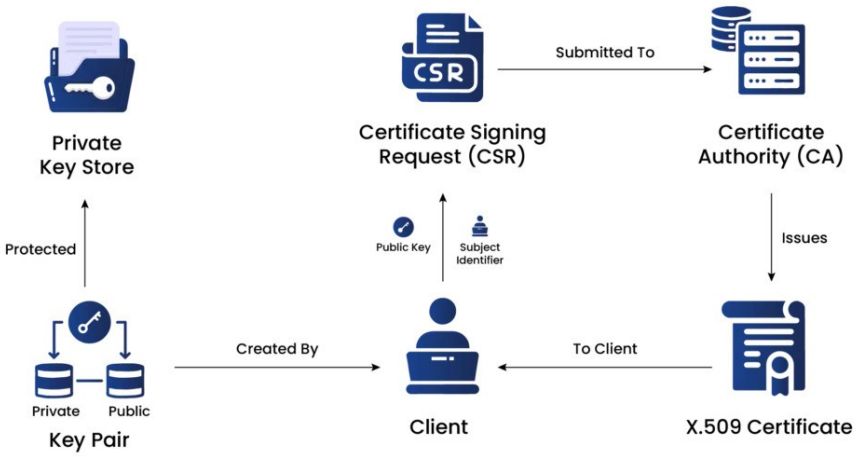
2.1. Basisbegrippen

2.1.1. Public key infrasctructure

Om het onderzoeksonderwerp samen met de achterliggende uitdaging te begrijpen, is het belangrijk om de werking van een Public Key Infrastructure (PKI) te begrijpen. Thales (z.d.) definieert een PKI als een set van hardware, software, policies, processen en procedures die noodzakelijk zijn voor het maken, beheren, uitgeven, gebruiken, opslaan en intrekken van digitale certificaten en publieke keys. PKI's zijn de basis die het gebruik van technologieëen zoals digitale handtekeningen en encryptie mogelijk maakt over een grote populatie van gebruikers. Zij helpen namelijk met het vaststellen van de identiteit van personen, apparaten en diensten, wat gecontrolleerde toegang tot systemen en bronnen alsook data beveiliging en controle mogelijk maakt.

2.1.2. Certificate authority en digitale certificaten

Een groot deel van PKI's zijn Digitale Certificaten en Certificate Authorities (CA's). Een certificate authority is een bedrijf of organisatie die de identiteit van entiteiten (zoals websites, e-mail adressen, bedrijven, individuen, enz.) valideert en ze vastbindt aan cryptografische sleutels door middel van het uitgeven van electronische documenten gekend als digitale certificaten. Een certificaat doet zich voor als een getuigschrift dat de identiteit valideert van de entiteit waaraan het is uitgegeven. (SSL Support Team, 2024)



Figuur 2.1: De stappen en componenten die deel uitmaken van het certificaat uitgave proces (Goel, 2024).

2.1.3. Aanvraagproces van certificaten

Wanneer een certificaat wordt aangevraagd bij een CA, moet de aanvrager eerst een public en private key genereren. De private key moet onder de controle en eigendom van de aanvrager blijven. In sommige gevallen worden de private keys gegenereerd en veilig bewaart in een Hardware Security Module (HSM). (SSL Support Team, 2024) Om de registratie van certificaten te starten, maakt de aanvrager een Certificate Signing Request (CSR) aan. Deze CSR bevat de public key en andere informatie van de aanvrager die in het certificaat zal worden opgenomen, zoals de domeinnaam voor een SSL/TLS certificaat of de aanvrager's e-mail adres voor een S/MIME certificaat.

Daarna wordt de CSR ingedient bij de CA. De CA zal de identiteit van de aanvrager samen met de bijkomende informatie verifiëren. De CA kan verschillende manieren gebruiken om de aanvrager zijn identiteit te verifiëren, zoals e-mail verificatie, domein validatie of manuele validatie van juridische documenten. Wanneer de CA het verificatie proces heeft voltooid en vaststelt dat de aanvrager legitiem is, zal het digitale certificaat worden uitgegeven. Het certificaat zal de aanvrager zijn public key en bijkomende informatie bevatten, alsook een geldigheidstermijn en de digitale handtekening van de CA.

Het uitgegeven certificaat wordt dan terug gebracht naar de aanvrager, afhankelijk van de CA en het certificaat type zal het certificaat geleverd worden op een verschillende manier zoals via e-mail, een beveiligd portaal of een andere methode. Na het ontvangen van het certificaat is het aan de aanvrager om het certificaat te installeren op de toepasselijke server of apparaat waar het zal worden gebruikt. Als

voorbeeld, een SSL/TLS certificaat wordt geïnstalleerd op een webserver voor een beveiligde connectie naar een website.

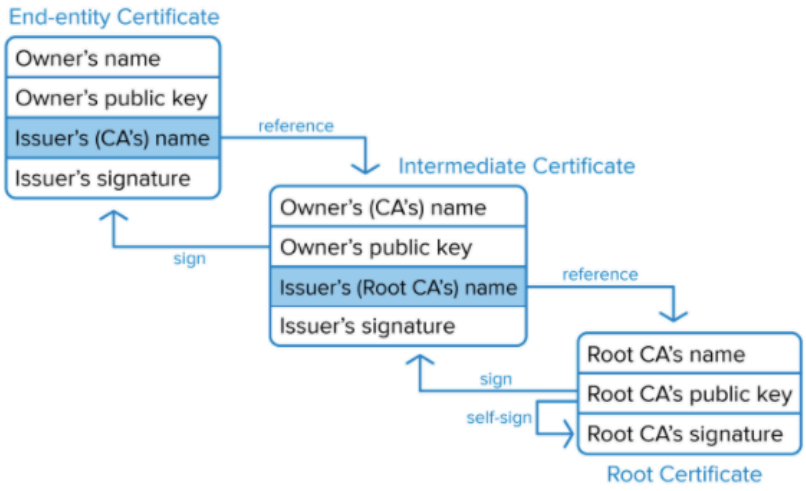
Eenmaal het certificaat is geïnstalleerd, kan het gebruikt worden voor protocolen die verantwoordelijk zijn voor beveiligde communicatie. Clients, gebruikers of andere entiteiten die in contact komen met de certificaat eigenaar kunnen de authenticiteit van het certificaat verifiëren aan de hand van de CA zijn digitale handtekening, wat een beveiligde en betrouwbare connectie verzekerd. Zoals eerder vermeld hebben deze certificaten een geldigheidstermijn (meestal 1 tot 2 jaren). Voor ze vervallen, moet de aanvrager het certificaat vernieuwen via een gelijkaardig proces om het te kunnen blijven gebruiken zonder onderbrekingen. (Goel, 2024)

2.1.4. Vertificatieproces van certificaten

Het verifiëren van een certificaat om de bepalen of het vertrouwd kan worden is belangrijk voor de veiligheid van de communicatie. Okta, Inc. (2023) zegt dat tijdens een SSL/TLS handshake, de client het certificaat van de server ontvangt. De client controleert of het certificaat nog niet vervallen is en dat de domeinnaam en IP adres op het certificaat gelijkaardig zijn aan dat van de server. Daarna zal de client kijken of het certificaat correct is ondertekend door een vertrouwde CA.

In de meeste gevallen zal de server certificate niet ondertekend zijn door de root CA die door de client wordt vertrouwd. In plaats van de root CA zal de client 1 of meerdere intermediate CA's vertrouwen zolang als hun ketting van vertrouwen terug leidt naar een root CA die de client vertrouwd.

Voor elke intermediate CA certificaat doet de client hetzelfde verificatie proces, waarbij de uitgever (issuer) overeen moet komen met de certificaat eigenaar van het volgende certificaat in de ketting. Ook wordt de digitale signatuur en public key van het certificaat bekeken om te kijken of deze correct is ondertekend. Dit proces herhaald zichzelf tot de client komt bij een self-signed root CA certificate die de client vertrouwd. Op dit moment heeft de client dan een cryptografische ketting van vertrouwen gemaakt tot de server en kan de SSL/TLS handshake verder gaan.



Figuur 2.2: Ketting van vertrouwen die de client nakijkt tijdens het verifiëren van een certificaat (Okta, Inc., 2023).

Bij dit verificatie proces eindigt de client steeds bij een root CA certificaat die door de client moet worden vertrouwd. Om te bepalen welke root CA's door de client worden vertrouwd bestaan er truststores.
Een truststore is een collectie van root certificaten die standaard worden vertrouwd door een client en worden beheerd door bedrijven die de client zijn operating system of browser ontwikkelen, zoals Microsoft, Mozilla en Google. Elke vendor heeft zijn eigen standaarden voor root certificaten maar ze vereisen allemaal dat een uitgevende CA een of meerdere controles ondergaan om hun vertrouwbaarheid, validiteit en conformiteit vast te stellen via de CA/B Forum Baseline Requirements vooraleer ze worden opgenomen in hun truststore. (Arampatzis, 2020)

2.1.5. Best-practices voor bedrijven hun truststores

Over al de truststores van de eerder vernoemde vendors zijn er heel wat certificaten die niet noodzakelijk zijn. Een studie van Perl e.a. (2014) toont dat alleen maar 66% van de certificaten in de truststore van Windows, Linux, MacOS, Firefox, iOS en Android noodzakelijk zijn voor het vertrouwen van websites. Dit zorgt ervoor dat de overige derde van de root certificaten in de truststores een potentieel veiligheidsrisico vormen voor de client.

Als oplossing hierop kunnen bedrijven het overwegen om deze standaard truststores af te wijzen. In de plaats daarvan kunnen ze best hun eigen aangepaste, corporate-level truststore maken en gebruik maken van certificate white-listing om te bepalen welke root certificates hierin kunnen opgenomen worden. Dit helpt bedrijven met het aanvals oppervlak te verkleinen door het limiteren van de hoeveel-

heid vertrouwde CA's en het markeren van niet-vertrouwde SSL/TLS sessies. Organisaties kunnen dan deze certificate whitelist en blacklist updaten op een regelmatige basis afhankelijk van benoodigdheden van hun evoluerende business requirements en groeiend CA landschap. (Arampatzis, 2020)

Het beheren van deze truststores kan een uitdaging zijn voor bedrijven, zeker bedrijven met heterogene netwerken (netwerken die clients hebben met verschillende operating systemen en browsers). Reddy en Wallace (2010) weerleggen dit door te zeggen dat deze trust anchors (Root CA certificaten) vaak bewaart worden in applicatie-specifieke of OS-specifieke truststores. Vaak heeft dan 1 machine een verschillend aantal truststores die niet gesynchroniseerd zijn met elkaar.

2.2. Verschillende truststores

Voor het implementeren van een oplossing voor deze uitdagingen is het van belang om te weten hoe de truststores van de verschillende besturingssystemen werken en hoe deze kunnen worden beheerd.

2.2.1. Windows truststore

Windows heeft een truststore die de “Trusted Root Certification Authorities store” heet. In de officiële documentatie van Microsoft vermeld Microsoft (2024a) dat de Trusted Root Certificate Authorities store op een Windows computer via de Microsoft Management Console (MMC) kan worden beheerd door de Certificate Manager snap-in te gebruiken. De Microsoft Management Console kan worden geopend door het commando “mmc.exe” uit te voeren in deen Run dialog. In het MCC venster kan men via file -> Add/Remove Snap-in de “Certificates” snap-in toevoegen.

Hierna kan men dan in het MCC venster onder “Certificates (local computer)” -> ”Trusted Root Certification Authorities”de lijst van vertrouwde root certificaten bekijken en beheren. Microsoft (2024a) vermeld ook dat standaard de Trusted Root Certificate Authorities store geconfigureerd is met een aantal publieke CA’s die de vereisten van de Microsoft Root Certificate Program hebben voltooid.

Naast de GUI manier van beheren van de truststore, biedt Microsoft ook een manier om het te beheren via de command line aan de hand van het “certutil” commando. Certutil.exe is een command-line programma geïnstalleerd als onderdeel van de Certificate Services. Certutil.exe kan gebruikt worden voor het tonen van certificate authority (CA) configuratie informatie, het configureren van Certificate Services en CA componenten te back-uppen en te herstellen. Dit programma kan ook certificaten, key pairs en certificate chains verifiëren. (Microsoft, 2025)

Om een certificate store te tonen aan de hand van certutil kan het volgende commando worden gebruikt:

```
1  certutil [options] -store [CertificateStoreName [CertId
    ↪  [OutputFile]]]
```

Waar CertificateStoreName de naam is van de certificate store, CertId de match token is van het certificaat en OutputFile de naam is van het bestand waarin de overeenkomende certificaten worden opgeslagen. (Microsoft, 2025)

Om een certificaat toe te voegen aan een certificate store kan het volgende commando worden gebruikt:

```
certutil [options] -addstore CertificateStoreName InFile
```

Waar CertificateStoreName de naam is van de certificate store en InFile de certificate file is die moet worden toegevoegd. (Microsoft, 2025)

2.2.2. Ubuntu truststore

Linux distributies hebben vaak een systeem truststore, mar vele Linux applicaties komen vaak met hun eigen truststores.

Binnen Ubuntu Server moet een certificaat in PEM formaat staan vooraleer het kan worden toegevoegd aan de truststore. Om een PEM-geformatteerd root CA certificaat met als voorbeeld naam ”local-ca.crt”te installeren in de truststore van Ubuntu Server kunnen de volgende commando's worden gebruikt:

```
sudo cp local-ca.crt /usr/local/share/ca-certificates
sudo update-ca-certificates
```

Het is hierbij belangrijk dat het certificaat bestand de extensie .crt heeft anders zal het niet worden verwerkt.

De truststore (die gegenereerd wordt door update-ca-certificates) is te vinden op de volgende locaties:

- Als een file (PEM bundel) in “/etc/ssl/certs/ca-certificates.crt”
- Als een OpenSSL-compatibele certificaat directory in “/etc/ssl/certs/ca-certificates.pem”

(Canonical Ltd., 2025)

2.2.3. RHEL truststore

Red Hat Enterprise Linux (RHEL) heeft een andere manier van het beheren van de truststore. RHEL biedt de Shared System Certificates aan. De Shared System Certificates storage laat NSS, GnuTLS, OpenSSL en Java toe om een standaard bron te delen voor het ophalen van systeem certificate anchors en black list informatie. Standaard bevat de truststore de Mozilla CA list, die positieve en negatieve trust informatie bevat. Het systeem laat toe om de Mozilla CA lijst aan te passen of om een andere CA lijst te gebruiken. (Red Hat, Inc., 2022)

In Red Hat Enterprise Linux 7 is de systeem-brede truststore te vinden in de directories “/etc/pki/ca-trust/” en “/usr/share/pki/ca-trust-source/”. De trust settings in

"/usr/share/pki/ca-trust-source/" worden behandeld met een lagere prioriteit dan de settings in "/etc/pki/ca-trust/" . Certificaat bestanden worden anders behandeld afhankelijk van de subdirectory waarin ze worden geplaatst:

- "/usr/share/pki/ca-trust-source/anchors/" of "/etc/pki/ca-trust/source/anchors/" : voor trust anchors.
- "/usr/share/pki/ca-trust-source/blacklist/" of "/etc/pki/ca-trust/source/blacklist/" : voor niet vertrouwde certificaten.
- "/usr/share/pki/ca-trust-source/" of "/etc/pki/ca-trust/source/" : voor certificaten in de extended BEGIN TRUSTED bestandsformaat.

(Red Hat, Inc., 2022)
Om een certificaat in de simpele PEM of DER bestandsformaten aan de lijst van vertrouwde CA's op het systeem toe te voegen, kan je simpelweg het certificaat bestand kopiëren naar de "/usr/share/pki/ca-trust-source/anchors/" of "/etc/pki/ca-trust/source/anchors/" directory. Om de systeem-brede truststore te updaten ka je het update-ca-trust commando gebruiken zoals volgt:

```
cp ~/certificate-trust-examples/Cert-trust-test-ca.pem
 ↪  /usr/share/pki/ca-trust-source/anchors/
update-ca-trust
```

Om de trust anchors op te lijsten, toe te voegen, veranderen of verwijderen kan het 'trust' commando gebruikt worden. Voor het oplijsten van de trust anchors wordt 'trust list' gebruikt.

Een trust anchor opslaan in de truststore kan met het 'trust anchor' sub-commando en het specifiëren van het pad (vb.: 'path.to') naar het certificaat bestand, zoals volgt:

```
trust anchor path.to/certificate.crt
```

Om een trust anchor te verwijderen kan een pad naar het certificaat of ID van het certificaat gebruikt worden:

```
trust anchor --remove path.to/certificate.crt
trust anchor --remove "pkcs11:id=%AA%BB%CC%DD%EE;type=cert"
```

(Red Hat, Inc., 2022)

2.2.4. Mozilla Firefox truststore

Applicaties kunnen ook hun eigen truststore hebben. Een voorbeeld hiervan is Mozilla Firefox. Firefox maakt gebruik van de NSS (Network Security Services) library voor het beheren van de truststore. De Network Security Services (NSS) library is een set van libraries ontwikkeld om cross-platform ontwikkeling van veilige client en server applicaties te ondersteunen. De libraries ondersteunen SSL v3, TLS, PKCS 5, PKCS 7, PKCS 11, PKCS 12, S/MIME, X.509 v3 certificaten en andere security standaarden. (Mozilla, z.d.) Firefox geeft bij initiatie een string met pad naar de directory waar NSS de security en configuratie data mag opslaan. NSS slaat 3 bestanden op in die directory:

- cert8.db: slaat publiek toegankelijke objecten op (Certificaten, CRL's, S/MIME records).
- key3.db: slaat private keys op.
- secmod.db: slaat de PKCS11 module configuratie op.

Als in deze directory grote security objecten zitten (zoals grote CRL's), zal NSS deze opslaan in bestanden in subdirectories genaamd 'cert8.dir'. In het geval dat cert8.db en/of key3.db niet bestaan, zal NSS de data lezen van oudere versies van deze databases (bv.: cert7.db, cert6.db,...) en zal deze data gebruiken om een nieuwe cert8.db en key3.db te maken. (Mozilla, 2018)

Ook beweert Mozilla (2024) dat standaard Firefox op Windows, MacOS en Android zal zoeken en gebruik maken van de third-party CA's die zijn opgenomen in de operating system zijn truststore. Firefox kan geconfigureerd worden om automatisch te zoeken naar CA's die in de Windows certificate store zijn toegevoegd door een gebruiker of administrator. Dit kan gedaan worden door de "security.enterprise_roots.enabled" optie in te stellen op "true" in de config van Firefox. Firefox zal dan de "HKLM/SOFTWARE/Microsoft/SystemCertificates" (Het pad dat overeenstemt met de API flag CERT_SYSTEM_STORE_LOCAL_MACHINE) registry directory inspecteren voor CA's die vertrouwd worden om certiifcaten uit te geven voor TLS web server authenticatie. (Mozilla, 2024)

Ook wordt er gekeken naar de volgende andere locaties:

- HKLM/SOFTWARE/Policies/Microsoft/SystemCertificates/Root/Certificates (pad in API flag CERT_SYSTEM_STORE_LOCAL_MACHINE_GROUP_POLICY)
- HKLM/SOFTWARE/Microsoft/EnterpriseCertificates/Root/Certificates (pad in API flag CERT_SYSTEM_STORE_LOCAL_MACHINE_ENTERPRISE)

Mozilla (2024) voorziet ook dat enterprise policies kunnen gebruikt worden voor het toevoegen van CA certificaten in Firefox. De "ImportEnterpriseRoots" key op

“true” zetten, zorgt ervoor dat Firefox root certificaten zal vertrouwen. De “Install” key zoekt standaard naar certificaten in de onderstaande locaties. Er kan ook een specifiek pad worden opgegeven. Als Firefox daar geen certificaten vindt zullen de standaard directories bekeken worden:

- Windows
  - %USERPROFILE%\AppData\Local\Mozilla\Certificates
  - %USERPROFILE%\AppData\Roaming\Mozilla\Certificates
- macOS
  - /Library/Application Support/Mozilla/Certificates
  - /Library/Application Support/Mozilla/Certificates
- Linux
  - /usr/lib/mozilla/certificates
  - /usr/lib64/mozilla/certificates

2.3. Bestaande tools

2.3.1. System Center Configuration Manager

Er bestaan verschillende gratis tools die kunnen helpen met het beheren van truststores over verschillende machines en applicaties. Deze tools zijn niet ontwikkeld specifiek voor het beheren van truststores maar bieden functionaliteiten aan voor het beheren van end-points hun configuratie. Voor Windows bestaat Microsoft System Center Configuration Manager (SCCM).

Microsoft (2024b) zegt dat Configuration Manager deel uitmaakt van de Microsoft Intune familie van producten. De Microsoft Intune familie van producten is een geïntegreerde oplossing voor het beheren van al jouw apparaten. Configuration Manager vebreed en werkt samen met vele Microsoft technologieëen en oplossing zoals onder andere Certificate Services.

2.3.2. Ansible

Een meer algemene oplossing die werkt voor beide Windows en Linux is Ansible. Ansible is een open-source IT automation engine die provisioning, configuratie management, applicatie deployment, orkestratie en vele andere IT taken automatiseert. Ansible kan gratis worden gebruikt en het project profiteert van de ervaring en intelligentie van zijn duizende gebruikers. (Red Hat, Inc., z.d.-a)

Ansible zijn grootste sterkte is eenvoud. Het heeft ook een sterke focus op beveiliging en betrouwbaarheid, met zo weinig mogelijk bewegende delen. Het gebruikt OpenSSH voor transport (met andere transport en pull modes beschikbaar als alternatieven), en gebruikt leesbare taal die ontworpen is om snel aan de start te kunnen gaan zonder enige training. Red Hat Ansible Automation Platform is een abbonement gebaseerde oplossing die bouwt op de basis van Ansible met een aantal ondernemingsgerichte functionaliteiten.

Beide community Ansible en Ansible Automation Platform zijn gebouwt op het concept van een control node en een managed node. Ansible wordt gebruikt op de control node waar bijvoorbeeld een gebruiker een ansible-playbook command uitvoert. Managed nodes zijn de apparaten die worden geautomatiseerd zoals bijvoorbeeld een Windows server. Voor Linux en Windows te automatiseren, connecteert Ansible met de managed nodes en verstuurt het kleine programma's genaamd Ansible modules uit. Deze programma's zijn geschreven als bronmodellen van de gewenste staat van het systeem. Ansible voert dan deze modules uit (standaard over SSH), en verwijdert ze na ze voltooien. Deze modules zijn ontworpen om idempotent te zijn waar mogelijk, zodanig ze alleen een aanpassing uitvoeren op een

systeem als dat nodig is. (Red Hat, Inc., z.d.-c)

2.3.3. Puppet

Een andere tool die mogelijks kan helpen is Open Source Puppet. Puppet is een open source configuratie management tool geproduceert door Puppet Labs. Er kunnen systeem configuraties gedefinieerd worden aan de hand van Puppet's declaratieve taal en deze configuraties kunnen dan opgeslagen worden in bestanden genaamd Puppet Manifests. Puppet kan standalone op een systeem draaien of in een agent/master configuratie. Met standalone Puppet draaien systemen het programma lokaal om hun eigen configuraties aan te passen. Met agent/master Puppet beheert een centrale Puppet master server of servers meerdere systemen die de Puppet agent draaien. Bij beide implementaties worden Puppet manifests gebruikt om systeem configuraties naar de verwachte staat te brengen. Daarnaast bestaat er ook nog een commerciële versie van Puppet genaamd Puppet Enterprise. (Red Hat, Inc., z.d.-b)

2.3.4. Salt

Naast Ansible en Puppet bestaan er ook nog andere tools zoals Saltstack en Chef. Gebouwt op Python, is Salt een event-driven automatiseringstool en framework dat is ontworpen om complexe IT systemen te deployeren, configureren en beheren. Salt kan gebruikt worden voor veel voorkomende administratie taken te automatiseren en te verzekeren dat alle componenten van je infrastructuur operationeel zijn in de verwachte staat. Salt heeft vele mogelijke gebruiksdoeleinden, zoals configuratie beheer, wat als volgt inhoud:

- Het beheren van het deployeren van besturingssystemen en hun configuratie.
- Het installeren en configureren van software applicaties en diensten.
- Het beheren servers, virtuele machines, containers, databases, web servers, netwerk apparaten en meer.
- Verzekeren dat de configuratie consistent is en het vermijden van drift.

Salt maakt het mogelijk om applicaties te implementeren en beheren die gebruik maken van elke technologiestack die op bijna elk besturingssysteem draait, inclusief verschilllende soorten netwerkapparaten zoals switches en routers van verschillende leveranciers. Daarnaast kan het ook gebruikt worden voor het automatiseren en orkestreren van routine IT processen zoals veel voorkomende noodzakelijke taken voor geplande server downtime of het upgraden van besturingssystemen en applicaties. Ook kan Salt gebruikt worden om een zelfbewuste, zelf herstellende

systemen te maken die automatisch kunnen reageren op uitvallingen, veel voorkomende administratie problemen of andere belangrijke gebeurtenissen. (VMware, Inc., 2025)

2.3.5. Chef

Progress Software Corporation (2024) noemt Chef een automation company. Vandaag biedt Chef automatiseringsoplossingen voor beide infrasctructuur en applicaties van ontwikkeling tot productie. Chef Infra automatiseert hoe infrastructuur is geconfigureerd, gedeployed en beheerd doorheen het netwerk, ongeacht de grootte.

Chef Workstation maakt het mogelijk om 'cookbooks' te schrijven en je infrastructuur te beheren. Chef Workstation hoort te draaien op de machine die je dagelijks gebruikt, ongeacht of deze Windows, Linux of MacOS draait. Eenmaal het ontwikkelen van je code gedaan is op je workstation, dan kan je deze code uploaden naar de Chef Infra Server. De Chef Infra Server is een centrale opslagplaats voor je configuratie data. Het slaat cookbooks, policies en metadata van elk systeem op. Het communiceren met de Chef Infra Server vanaf een workstation kan via het 'knife' commando. Chef Infra Clients contacteren deze Chef Infra Server periodiek om de laatste cookbooks op te halen. Als de huidige staat van de node niet overeenstemt met wat de cookbook beschrijft, zal de Chef Infra Client de cookbook instructies uitvoeren. (Progress Software Corporation, 2024)

2.3.6. Hashicorp Vault

Naast de automatiseringstools kan een centrale opslagplaats voor de certificaten ook mogelijks helpen met het beheren van de certificaten die in de truststores terecht moeten komen.

Hashicorp Vault biedt een gecentraliseerde, goed gecontrolleerde toegangsbeveiliging en beheer van geheime en doelskritieke gegevens aan, ongeacht van de infrastructuur die wordt gebruikt. Deze tool helpt om geheimen data, zoals credentials, encryptie sleutels, authenticatie certificaten en andere kritische stukken van informatie centraal te beheren. (Hashicorp, Inc., z.d.)

2.4. Infrastructuur studie

Voor een realistisch beeld te scheppen van een bedrijfsomgeving, zal er gekeken worden naar resultaten van meerdere vragenlijsten en enquêtes die het marktaandeel van verschillende technologieëen onderzoeken.

2.4.1. Webservers

Netcraft LTD. (2025) publiceert maandelijks een web server survey waarin ze de marktaandelen van web servers onderzoeken, uit de resultaten van de survey van januari 2025 blijkt Nginx het grootste marktaandeel te hebben binnen alle onderzochte sites met 19,60% alsook alle onderzochte actieve sites met 18,89%. Naast Nginx heeft Apache het tweede grootste marktaandeel met 16,96% van alle onderzochte sites en 17,24% van alle actieve sites.

2.4.2. Automatiseringstools

Binnen de Stack Overflow developer survey van 2024 kan er ook gekeken worden naar veel gebruikte technologieëen binnen de professionele wereld. In de resultaten kan er gekeken worden naar welke van de eerder vermelde tools het meest gebruikt worden in de professionele wereld. (Stack Exchange Inc., 2024) rapporteert onder overige tools dat Ansible gebruikt wordt door 8,1% van de ondervraagde professionele developers, Pupper door 1,1% en Chef door 0,7%.

2.4.3. Databases

Binnen de resultaten van de Stack Overflow survey kan er ook gekeken worden naar de meest gebruikte databases. Uit deze resultaten blijkt dat PostgreSQL de meest gebruikte database is met 51,9% van de ondervraagde professionele developers die het gebruiken. MySQL is de tweede meest gebruikte database met 39,4% van de ondervraagde professionele developers die het gebruiken. (Stack Exchange Inc., 2024)
Deze informatie zal gebruikt worden voor het bepalen van de gebruikte technologieëen binnen de proof-of-concept virtuele omgeving.

3

Methodologie

Het onderzoek naar centrale truststorebeheer zal opgedeeld worden in drie fases: een literatuurstudie, praktijkstudie en uiteindelijke rapportage van de configuraties die gebruikt werden.

In de eerste anderhalve maand zal de eerste fase van het onderzoek uitgevoerd worden, de literatuurstudie.
Het doel in deze fase is om een diepgaand begrip te krijgen over trust management, certificaatbeheer en de verschillende aanpakken bij truststorebeheer.
In deze literatuurstudie zal er gekeken worden naar de bestaande theorieën, technologieën en andere tools die in de realiteit worden gebruikt.
Ook zal er gekeken worden naar de uitdagingen van trust management in een omgeving met netwerksegmentatie en diverse besturingssystemen.
Doorheen deze fase zal er periodiek overleg zijn met de co-promotor om de voortgang te bespreken en een mogelijkse oplossing te kiezen die verder kan worden uitgewerkt in de praktijkstudie.

De info die verkregen wordt in deze literatuurstudie zal de basis leggen voor de volgende fase van dit onderzoek: de praktijkstudie.
Deze fase zou een totale tijdsduur van 6 weken hebben. Deze praktijkstudie opgedeeld in 3 delen.
het ontwerpen van een virtuele omgeving, de implementatie van het centraal truststorebeheer en de evaluatie van de oplossing.

In de eerste stap zal er weer worden gekeken naar veel gebruikte software en besturingssystemen binnen bedrijven om zo een realistische infrastructuur te creëren.
Ook wordt er een certificate authority (CA) opgezet om het beheer van certificaten en een interne private PKI te simuleren.

Deze virtuele omgeving zal worden opgezet binnen GNS3, een netwerkemulator die het mogelijk maakt om complexe netwerken te simuleren.
De volledige opzet van het netwerk zal een week duren.

De tweede stap, de implementatie van een centraal trust management systeem, begint eind maart en zal vijf weken duren.
In deze fase wordt een gecentraliseerde oplossing voor trust management opgezet. Dit gebeurt door middel van de tools die gekozen werden na de literatuurstudie in combinatie met eventuele zelfgeschreven scripts voor het certificaatbeheer.
Het doel hier is dat bij het toevoegen of verwijderen van een nieuw root certificaat, de truststores op elk systeem in het netwerk die binnen deze laag trust valt dat root certificaat bevat.

Tijdens de implementatie vindt ook de derde stap van de praktijkstudie plaats, die bestaat uit de evaluatie en validatie van de oplossing.
Deze fase begint na het succesvol implementeren van een oplossing.
De centrale vraag is hoe kwaliteitsvol de oplossing is en of deze voldoet aan de verwachtingen en ook om te kijken wat de oplossing niet heeft kunnen bereiken.
Dit wordt bepaald door testcases op te stellen om de functionaliteiten te testen, zoals verwijderen of toevoegen van root certificaten binnen bepaalde netwerksegmenten.
Ook wordt de schaalbaarheid van de oplossing in vraag gesteld.
Bij de schaalbaarheid zal er gekeken worden naar hoe de oplossing omgaat met het veranderen of uitbreiden van het netwerk alsook de groei van het aantal root certificaten.
Hierna wordt ook nagedacht over hoe de tekortkomingen van de oplossing kunnen worden opgelost of welke alternatieven er zijn om deze tekortkomingen te vermijden.

Als laatste fase van dit onderzoek zal er een rapportage gemaakt worden binnen deze paper onder het hoofdstuk ‘Proof-of-concept’. Dit zal een tijdsduur van 4 weken hebben. De rapportage zal de configuraties van de oplossingen binnen de PoC beschrijven alsook de positieve en negatieve punten van deze oplossing.
Op basis van de bevindingen worden dan conclusies en aanbevelingen geformuleerd voor de implementatie van centraal truststorebeheer alsook welke aspecten mogelijks verder onderzoek vereisen.

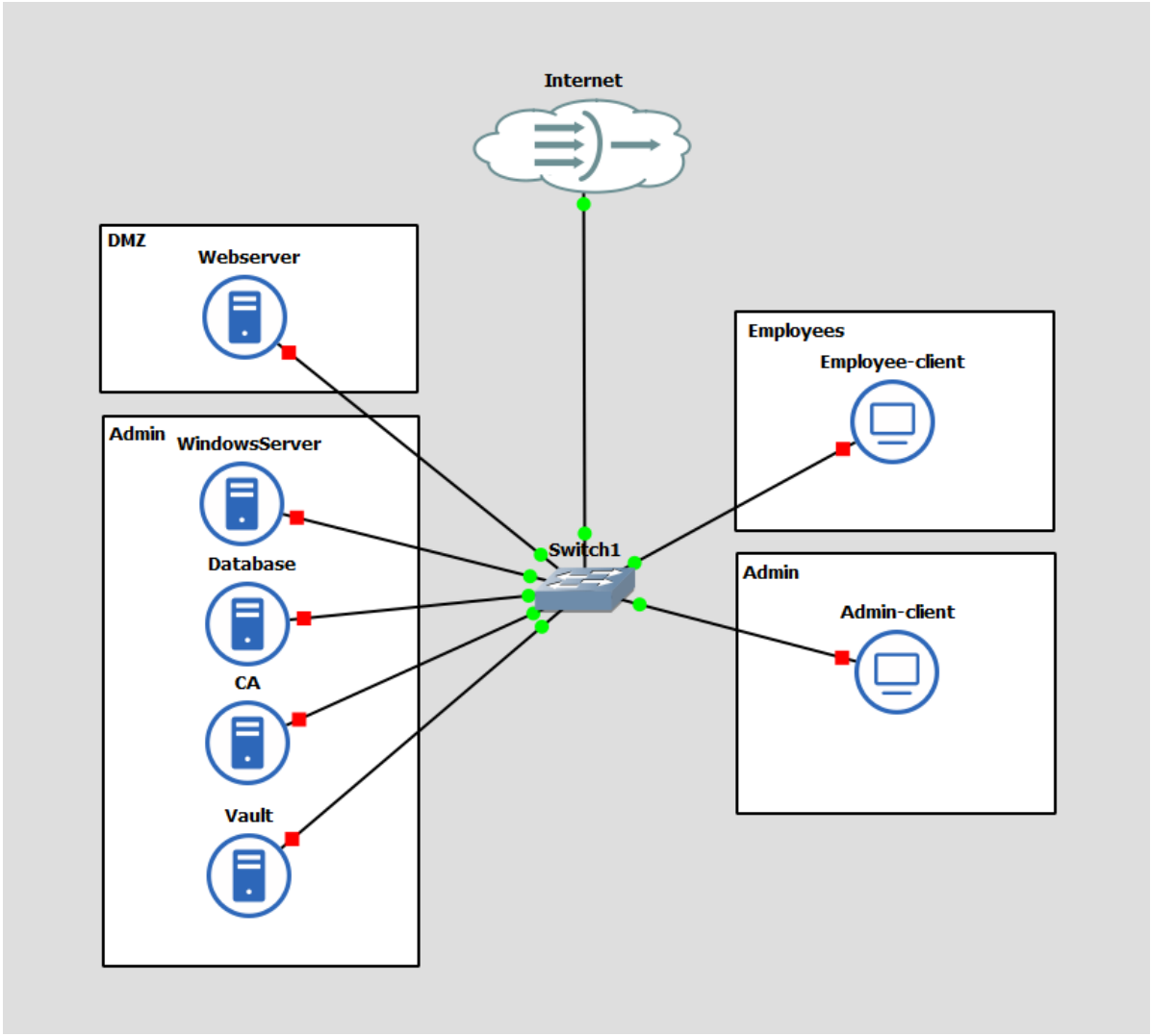
4

Proof-of-concept

4.1. Gekozen infrastructuur

Voor het opzetten van een oplossing en deze te kunnen testen en evalueren, is er een virtuele omgeving opgezet. Deze virtuele omgeving is opgezet in GNS3, een netwerkemulator die het mogelijk maakt om netwerkapparatuur alsook virtuele machines te simuleren. Binnen deze virtuele omgeving zal er aan netwerksegmentatie gedaan worden, het netwerk zal opgedeelde worden in 3 segmenten: een Admin, DMZ en employee segment.

Op basis van de ondervindingen uit de infrastructuurstudie werd er gekozen om de volgende besturingssystemen en software te gebruiken:



Figuur 4.1: De opzet van de proof-of-concept omgeving.

Naam Rollen	Netwerksegment	OS
Webserver (Webserver, Chef infra client)	DMZ	Ubuntu 22.04
WindowsServer (Active directory, DNS server, SCCM server)	Admin	Windows server 2022
Database (Chef workstation, Chef infra client)	Admin	Almalinux 8.8
CA (Ansible control node, Chef infra server)	Admin	Almalinux 8.8
Vault (Vault server)	Admin	Ubuntu 22.04
Employee-client (SCCM Agent)	Employee	Windows 11 Enterprise
Admin-client (SCCM Agent)	Admin	Windows 11 Enterprise

Tabel 4.1: Tabel met alle machines in de proof-of-concept omgeving.

4.1.1. CA

De CA server draait een Almalinux 8.8 image en is een Certificate Authority (CA) die draait in het admin netwerksegment, deze CA zal de rol van interne CA spelen. In de infrastructuurstudie werd geen informatie gevonden rond software voor het opzetten van CA, hierdoor werd er gekozen om OpenSSL wegens bestaande kennis van deze software. Met OpenSSL werd een root certificaat getekend door de CA server zelf, naast de self-signed certificate werd er ook een leaf certificate gemaakt worden voor de webserver binnen deze omgeving. Deze server zal ook de Ansible control node zijn voor de eerste oplossing, voor de tweede gevonden oplossing zal deze de rol van de Chef infra server innemen.

4.1.2. Webserver

De webserver draait op Ubuntu 22.04 en heeft een Nginx webserver draaien met een certificaat dat ondertekend werd door de eerder vermelde CA server. Dit is ook de enigste server die in het DMZ netwerksegment staat. Het doel van deze webserver is om de functionaliteit van het aanpassen van de truststore te testen, als een end-point de webpagina kan bereiken zonder waarschuwing via het https protocol, dan betekent dat de server “CA” zijn root certificaat aanwezig is in de end-point zijn truststore. Daarnaast zal deze servero ok gebruikt worden om de functionaliteiten van een gevonden oplossing te testen binnen het DMZ netwerksegment.
De server zal later ook de rol van Chef infra client spelen om te kunnen communiceren met de Chef infra server, daarnaast draait ook Hashicorp Vault Agent op deze server voor communicatie met de Vault server in de tweede oplossing.

4.1.3. Windows server

Deze server draait Windows Server 2022 en is de domein controller van de virtuele omgeving, gesitueerd in het admin netwerksegment. Het opgezette domein is “Bachelorproef.local” en de server heeft een Active Directory (AD). De Active Directory bevat 2 logins voor de Windows clients “Employee-client” en “Admin-client”. Binnen de AD zal elke Windows client die het netwerk betreedt automatisch gestoken worden. Deze clients worden manueel in de juiste organisational unit (OU) gestoken. In dit geval zijn dit de OU “Employee” en “Admin”. De rol van DNS server is ook opgenomen in deze server. Ondanks dat de infrastructuurstudie aangaf dat er populairdere DNS servers zijn, werd er gekozen om de ingebouwde DNS server van Windows Server te gebruiken, zodanig alle domain services op een centraal punt beheerd kunnen worden. De Windows server heeft ook SCCM (System Center Configuration Manager) server draaien die later in het onderzoek voor de tweede oplossing gebruikt zal worden om de truststores van de Windows clients te beheren.

4.1.4. Database

Deze server draait Almalinux 8.8 en ligt ook binnen het admin segment, ondanks de naam heeft deze server geen database draaien, het originele idee was om de PostrgeSQL database voor de Chef infra server op deze server te draaien, maar dit werd later aangepast naar de CA server. Aangezien deze server geen cruciale rollen heeft werd deze gebruikt als end-point waarop de functionaliteiten van de oplossingen binnen het Admin netwerksegment getest kunnen worden. Daarnaast zal deze server ook de rol van Chef workstation hebben om de Chef infra server en clients te kunnen beheren.

4.1.5. Vault

De Vault server draait ook Ubuntu 22.04 en heeft een Hashicorp Vault server draaien. Dit is ook de laatste server die in het admin netwerksegment staat. Deze Vault server zal gebruikt worden in de tweede oplossing om de root certificaten op een centrale plaats op te slaan.

4.1.6. Employee-client

Deze Windows client draait Windows 11, en bevindt zich in het Employee netwerksegment. Deze client maakt deel uit van het domein “Bachelorproef.local” en is ook lid van de OU “Employee”. Voor de tweede oplossing zal deze ook SCCM agent draaien.

4.1.7. Admin-client

Deze Windows client draait ook Windows 11 en bevindt zich in het Admin netwerksegment. Deze client maakt ook deel uit van het domein “Bachelorproef.local” en is ook lid van de OU “Admin”. Deze client draait ook SCCM agent bij de tweede oplossing.

4.2. Voorbereiding truststores

Zoals ondervonden in de literatuurstudie, is het belangrijk om onnodige root certificaten te verwijderen uit de default truststores die bij installatie van de besturingssystemen zijn meegeleverd. Binnen deze proof-of-concept omgeving zijn er 3 verschillende systeem truststores, 1 voor de Windows end-points en 1 voor Ubuntu en Almalinux end-points. Het is mogelijks een goed idee om eerst deze truststores te back-uppen vooraleer deze aan te passen, moest een probleem optreden op een later moment kan de truststore teruggezet worden naar de originele staat.

In Windows zijn de root certificaten opgeslagen in de “Trusted Root Certification Authorities” store. Om certificaten te verwijderen uit deze store, kan er gebruik gemaakt worden van de “certlm.msc” tool of het volgende PowerShell commando

met als voorbeeld een root certificaat met de naam “untrusted”:

```
Get-ChildItem -Path Cert:\LocalMachine\Root | Where-Object {
 ↪  $_.Subject -like "*untrusted*" } | Remove-Item
```

Ubuntu werkt met een config file die zich bevindt in de directory “/etc/ca-certificates.conf”. Deze file bevat een lijst van root certificaten die vertrouwd worden door het systeem. Om root certificaten te verwijderen uit deze file, kunnen deze lijnen verwijderd of gecommentarieerd worden. Wanneer het commando “update-ca-certificates” uitgevoerd wordt, worden de root certificaten die in deze file staan, samen met de root certificaten die in de directory “/usr/local/share/ca-certificates” staan, samengevoegd in de truststore van het systeem. Deze truststore is simpelweg een file die de bundle met root certificaten bevat. Dit bestand kan gevonden worden in de directory “/etc/ssl/certs/ca-certificates.crt”.

Na het aanpassen van de config file kan het ca-certificates.crt bestand verwijderd worden en kan het commando “update-ca-certificates” uitgevoerd worden om de truststore opnieuw aan te maken als volgt:

```
sudo rm /etc/ssl/certs/ca-certificates.crt
sudo update-ca-certificates --fresh
```

Het is hierbij dan ook belangrijk om te weten dat de certificaten uit /usr/local/share/ca-certificates toegevoegd worden aan deze truststore, deze directory is standaard leeg, moest hier een certificaat staan, zal dit certificaat ook toegevoegd worden aan de truststore.

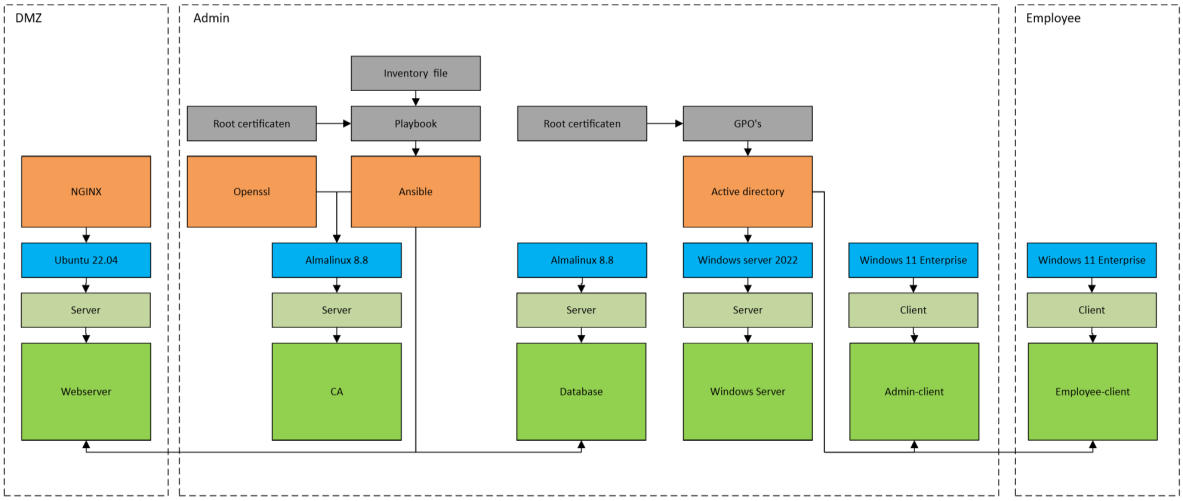
In Almalinux zitten de root certificaten in verschillende bundle files in directories per applicaties. Deze directory is “/etc/pki/ca-trust/extracted/”, binnenin deze directory liggen dan meerdere subdirectories benaamd naar de bijhorende applicatie die elk een bundle file bevatten, hieruit moeten de onnodige certificaten verwijderd worden.

Om deze aan te passen moet er eigenlijk maar slechts 1 bestand aangepast worden. De file /etc/pki/tls/certs/ca-bundle.crt bevat de root certficaten die standaard vertrouwd worden en deze inhoud wordt bij de uitvoer van het commando “update-ca-trust” toegevoegd aan de bundle file in elke subdirectory. Elk certificaat heeft zijn naam in commentaar vorm staan, om een certificaat te verwijderen kan de lijn met de CA naam tot en met de eerst volgende lijn met “—END CERTIFICATE—” verwijderd worden. Zoals bij ubuntu is het belangrijk om te weten dat als het commando “update-ca-trust” uitgevoerd wordt, de root certificaten die in de directory

“/etc/pki/ca-trust/source/anchors/” staan ook toegevoegd worden aan alle truststores van al de applicaties. Deze zou normaal ook standaard leeg moeten zijn.

Voor de gevonden oplossingen werden alle certificaten uit de default truststores verwijderd, alle nodige certificaten kunnen via de oplossingen terug toegevoegd worden aan de truststores. Dit zorgt ervoor dat de default truststores nooit meer aangeraakt moeten worden en enkel de certificaten binnen de oplossing zelf moeten beheerd worden. Een belangrijke observatie tijdens het uitvoeren van dit onderzoek was dat zowel Windows als Linux minimum 1 certificaat moeten hebben in hun truststore, Windows zal het verwijderen van het allerlaatste certificaat niet toelaten en Applicaties binnen Linux zullen foutmeldingen geven wanneer hun bijhorende bundle file niet bestaat en/of leeg is.

4.3. Eerste oplossing



Figuur 4.2: Het architectuurplan van de werking van de eerste oplossing.

De initieel gevonden oplossing bestond uit het gebruik van Ansible voor de Linux end-points en Group Policy Objects (GPO's) voor de Windows end-points. Beide de Windows server en CA server (die de Ansible control server is) hebben een kopie van de root certificaten die bij de Windows kant in de GPO's worden gestoken en bij de Linux kant overgezet worden via Ansible.

4.3.1. Oplossing door middel van GPO's met root certificaten

Om Windows clients hun truststores centraal te kunnen beheren, kan er een GPO (Group Policy Object) aangemaakt worden die root certificaten bevat en deze kan dan toegepast worden op een organizational unit (OU) binnen de Active Directory. Er kan dus een GPO aangemaakt worden per OU (in dit geval de 3 OU's voor de 3 netwerksegmenten) die elk de juiste root certificaten bevatten.

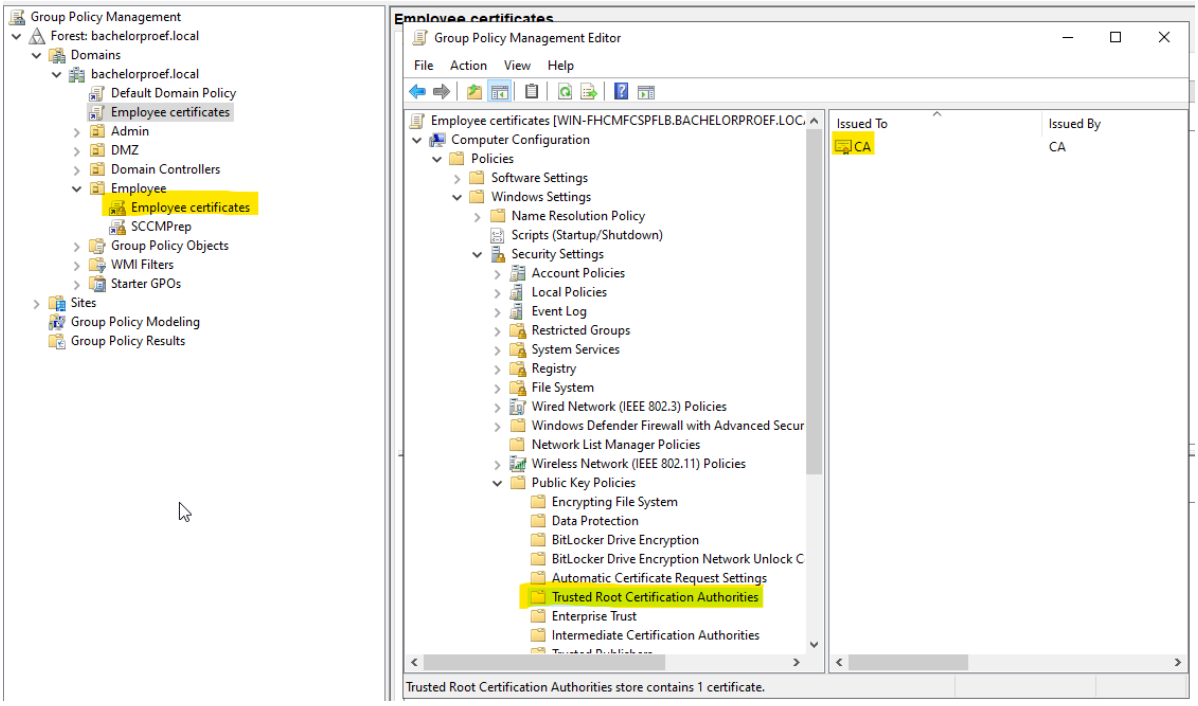
Voor deze proof of concept zal er een verzameling van root certificaten opgeslagen worden in een directory op de Windows server, deze directory zal 3 subdirectories bevatten, 1 voor elk netwerksegment. De root certificaten die in deze subdirectories staan zijn de root certificaten die de clients in dat netwerksegment moeten vertrouwen. De GPO's zullen dan de root certificaten uit deze subdirectories halen en deze toevoegen aan de truststore van de clients in dat netwerksegment.

De GPO's kunnen als volgt aangemaakt worden:

- Open de Group Policy Management Console (GPMC) op de Windows server.
- Maak een nieuwe GPO aan door met de rechtermuisknop op de OU te klikken en "Create a GPO in this domain, and Link it here" te selecteren.

- Geef de GPO een naam, bijvoorbeeld “Employee certificates”.
- Klik met de rechtermuisknop op de GPO en selecteer “Edit”.
- Ga naar Computer Configuration > Policies > Windows Settings > Security Settings > Public Key Policies > Trusted Root Certification Authorities.
- Klik met de rechtermuisknop en selecteer “Import”.
- Volg de wizard om het root certificaat te importeren vanuit de directory op de Windows server.
- Herhaal deze stappen voor de andere OU’s en root certificaten.

De GPO’s kunnen dan toegepast worden, dit kan door met de rechtermuisknop op de OU te klikken en “Link an Existing GPO” te selecteren. Selecteer dan de GPO die je wilt toepassen op de OU. Daarna moet de policy nog enforced worden door met de rechtermuisknop op de GPO te klikken en “Enforce” te selecteren. Dit zorgt ervoor dat de GPO ook toegepast wordt op de OU en alle sub-OU’s die binnen deze OU zitten.
Voor de GPO “Employee certificates” zou dit er als volgt uitzien:



Figuur 4.3: De GPO “Employee certificates” in het GPMC.

Op de clients kan je dan de GPO’s toepassen door een reboot of door het volgende commando uit te voeren in de command prompt:

```
1    gpupdate /force
```

4.3.2. Oplossing voor Linux end-points met Ansible

Om het probleem van het centraal beheren van de root certificaten op te lossen, kan er voor de Linux end-points een Ansible playbook gemaakt worden dat root certificaten kan toevoegen aan de truststores. De CA server zal in deze proof-of-concept ook de Ansible control server zijn. Dit is de server die de Ansible playbooks zal uitvoeren op de end-points via SSH connecties. Hiervoor is een inventory bestand nodig dat de IP adressen of domein naam van de end-points bevat samen met de gebruiker die gebruikt zal worden voor de SSH verbinding en de waar de public key die gebruikt wordt door de Ansible node ligt. Dit bestand voor deze omgeving ziet er als volgt uit:

```
[DMZ]
web ansible_host=bpweb.bachelorproef.local
  ↪ ansible_user=ubuntu
  ↪ ansible_ssh_private_key_file=/home/almlainux/.ssh/id_rsa

[Admin]
db ansible_host=db.bachelorproef.local ansible_user=almalinux
  ↪ ansible_ssh_private_key_file=/home/almalinux/.ssh/id_rsa
ca ansible_host=ca.bachelorproef.local ansible_user=almalinux
  ↪ ansible_ssh_private_key_file=/home/almalinux/.ssh/id_rsa
```

Codefragment 4.1: Het gebruikte inventory bestand voor Ansible.

Net zoals bij de Windows server zal er een directory aangemaakt worden op de server “CA” die subdirectories bevat voor elk netwerksegment. Deze subdirectories bevatten de root certificaten die de end-points in dat netwerksegment moeten vertrouwen. Met deze bestanden en directories kan er een Ansible playbook gemaakt worden dat de root certificaten uit de juiste subdirectory haalt en toevoegt aan de truststore van de end-points. Dit kan gedaan worden met het volgende Ansible playbook:

```yaml
- name: truststore update
  hosts: all
  become: true
  vars:
    cert_source_dir: >-
      {{ '/home/almalinux/Ansible/DMZ' if 'DMZ' in group_names else
       ↪  '/home/almalinux/Ansible/Admin' }}
    dest_cert_dir: >-
      {{ '/usr/local/share/ca-certificates' if ansible_os_family ==
       ↪  "Debian" else '/etc/pki/ca-trust/source/anchors' }}
    update_command: >-
      {{ 'update-ca-certificates' if ansible_os_family == "Debian"
       ↪  else 'update-ca-trust extract' }}

  tasks:
    ### LEDIGEN DIRECTORIES ###
    - name: verwijderen van truststore directory
      file:
        path: "{{ dest_cert_dir }}"
        state: absent
      when: ansible_os_family in ["Debian", "RedHat"]

    - name: de truststore directory opnieuw aanmaken
      file:
        path: "{{ dest_cert_dir }}"
        state: directory
        owner: root
        group: root
        mode: '0755'
      when: ansible_os_family in ["Debian", "RedHat"]

    ### CERTIFICATEN INSTALLEREN ###
    - name: Kopieer certificaten van Ansible control node naar de
     ↪  truststore directory op end-point
      copy:
        src: "{{ cert_source_dir }}/"
        dest: "{{ dest_cert_dir }}/"
        owner: root
        group: root
        mode: '0644'
        remote_src: no

    ### TRUST STORE UPDATEN ###
    - name: Update systeem truststore
      command: "{{ update_command }}"
```

Codefragment 4.2: De Ansible playbook die certificaten van de Ansible control node naar de end-point truststore directory kopieert.

```
    ### DEBUG ###
    - name: Bevestiging
      debug:
        msg: "Alle truststores zijn up-to-date"
```

Codefragment 4.2: De Ansible playbook die certificaten van de Ansible control node naar de end-point truststore directory kopieert.

Het playbook begint met het aanmaken van 3 variabelen, de eerste variabele genaamd “cert_source_dir” bevat de directory waar de root certificaten staan die de end-point moet vertrouwen, deze directory is afhankelijk van de groep waar de end-point in zit binnen de inventory file. Als tweede variabele “dest_cert_dir” wordt de directory bepaald waar de root certificaten naartoe gekopieerd worden, deze directory is afhankelijk van de linux distributie van de end-point. De laatste variabele “update_command” bevat het commando dat gebruikt zal worden om de truststore te updaten, dit is ook afhankelijk van de linux distributie.

Deze playbook zal hierna de truststore directory ledigen door het te verwijderen, dit is opzich geen probleem want alle certificaten zijn momenteel nogsteeds aanwezig in de truststore bundle files van de applicaties. Vervolgens zal deze directory weer aangemaakt worden met de juiste permissies en hierna zullen de root certificaten uit de juiste bron directory gekopieerd worden naar de juiste doel directory. Als laatste zal de truststore geüpdatet worden met het correcte commando afhankelijk van de linux distributie. Hierbij zullen de root certificaten in de bundle files van de applicaties gestoken worden.

Een belangrijke opmerking bij deze playbook is dat het alleen zal uitgevoerd worden voor Ubuntu en Red Hat end-points, de stappen zullen overgeslagen worden voor andere distributies. Voor andere distributies moet er een juiste aanwijzing van variabelen voorzien worden alsook moeten deze vermeld worden onder elke stap in de playbook binnen het ’when’ veld.

Om het voorgaande Ansible playbook uit te voeren, kan er gebruik gemaakt worden van de volgende commando’s:

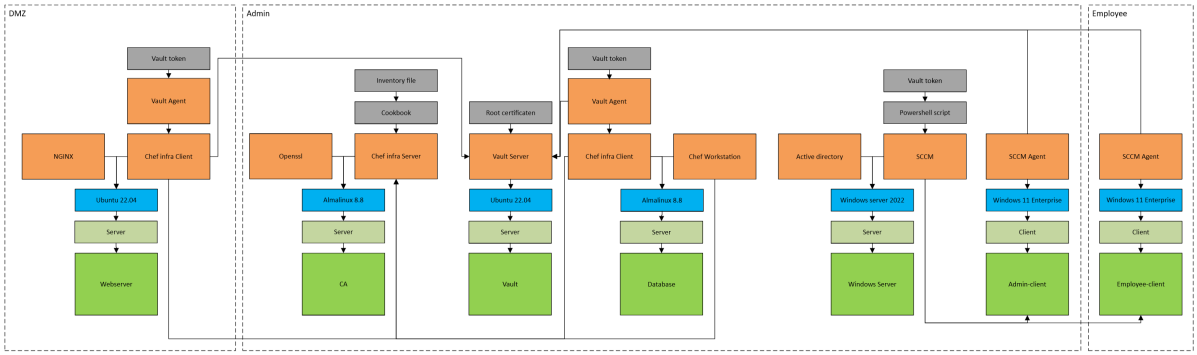
```
    ansible-playbook -i inventory playbook.yml
```

Waarbij “inventory” het inventory bestand is dat de IP adressen van de end-points bevat en “playbook.yml” het bovenstaande Ansible playbook is.

4.3.3. Pro's en Con's van de eerste oplossing

- Pro's:
  - De root certificaten kunnen eenvoudig beheerd worden binnen de Ansible node en Windows server directory.
  - Geen nodige installatie van extra software op de end-points.
  - End-points ondervinden geen hinder van de installatie van de root certificaten.
  - Er zijn weinig points of failure aangezien de oplossing geen bijkomende servers of software nodig heeft.
- Con's:
  - De GPO's passen alleen maar toe wanneer de end-points opnieuw opgestart worden of de GPO's geforceerd worden.
  - Er is geen directe functie voor het importeren van certificaten in een GPO via Powershell, dit betekent dat als er veel root certificaten zijn, het handmatig aanmaken van de GPO's tijdrovend zal zijn.
  - De certificaten staan in directories die niet beveiligd zijn, dit brengt een risico mee dat de certificaten binnen de directory kunnen worden aangepast of verwijderd.
  - Zowel de Windows server als de CA server (de Ansible control server) hebben een kopie van de root certificaten, bij een aanpassing van de root certificaten moeten deze op beide servers aangepast worden.
  - De Ansible playbook is niet schaalbaar, een server maakt SSH verbindingen met de end-points en voert de playbook uit op elke end-point. Dit betekent dat als er veel end-points zijn, de Ansible playbook traag zal zijn en mogelijk niet alle end-points kan bereiken.

4.4. Tweede oplossing



Figuur 4.4: Het architectuurplan van de werking van de tweede oplossing.

De tweede oplossing streeft ernaar de negatieve punten van de eerste oplossing te verhelpen. In de tweede oplossing zal er een Hashicorp Vault server gebruikt worden om root certificaten centraal op te slaan en deze te beheren. Hiervoor zal op de Linux end-points een Hashicorp Vault agent moeten draaien voor toegang te krijgen tot de Vault server met een token, op de Windows server zal er centraal een token aanwezig zijn voor alle Windows end-points. Aan de Windows kant van de oplossing zal er een System Center Configuration Manager (SCCM) server gebruikt worden om een Powershell script te deployen op de end-points die de juiste root certificaten uit de Vault server haalt en deze toevoegt aan de truststore. Voor de Linux end-points zal er hier in de plaats van Ansible een Chef infra server gebruikt worden die de instructies aan end-points aanbied om de certificaten uit de Vault server te halen en deze toe te voegen aan de truststore.

4.4.1. Installeren van een Vault server

Om het probleem van het dubbel beheren van de root certificaten op te lossen, moet er een centraal punt op het netwerk zijn die de root certificaten bevat. Dit kan gedaan worden door een Hashicorp Vault server op te zetten. Deze server brengt naast het beheren van de root certificaten ook autorisatie met zich mee waardoor de certificaten ook beveiligd zijn.
Binnen deze proof-of-concept omgeving is er een standalone server voorzien voor de Hashicorp Vault genaamd “Vault”. Deze server heeft een kopie van de directories met root certificaten die op de servers “CA” en “Windows Server” stonden. Daarnaast zal Vault in “dev” mode draaien, dit is een testmodus die het mogelijk maakt om snel te experimenteren met Vault zonder dat er een volledige configuratie nodig is. In deze modus zal de vault server ook geen persistentie hebben, wat betekent dat alle gegevens uit de vault verloren gaan als de vault sluit of de server stopt.

De vault server wordt opgestart met de volgende commando:

```
1     vault server -dev -dev-listen-address="0.0.0.0:8200"
```

Dit maakt de Vault server toegankelijk op all zijn IP adressen van de server op poort 8200.
De Vault server zal onderverdelingen hebben per netwerksegment, binnen elk netwerksegment zullen de root certificaten staan. Naast de root certificaten zal er ook een manifest binnen elk netwerksegment bestaan die een oplijsting bevat van zijn root certificaten. Deze manifest kan gebruikt worden door end-points om elk root certificaat binnen hun netwerksegment op te halen.

Voor het aanmaken van de indelingen en manifests werd er een bash script gemaakt die de root certificaten uit de directories haalt en deze toevoegt aan de vault. Dit script ziet er als volgt uit:

```
#!/bin/bash

VAULT_ADDR="http://127.0.0.1:8200"
export VAULT_ADDR

BASE_CERT_DIR="/home/ubuntu/"
SEGMENTS=("DMZ" "Admin" "Employee")

for SEGMENT in "${SEGMENTS[@]}"; do
    SEGMENT_DIR="$BASE_CERT_DIR/$SEGMENT"
    VAULT_PATH="secret/certs/$SEGMENT"

    echo "certificaten uploaden voor segment: $SEGMENT"

    # Remove the existing manifest and its metadata
    echo " - verwijderen van manifest en metadata voor $SEGMENT"
    vault kv delete "$VAULT_PATH/_manifest" || echo "geen manifest te verwijderen"
    vault kv metadata delete "$VAULT_PATH/_manifest" || echo "geen metadata te verwijderen"

    # certificaten uploaden van de directory
    for cert_file in "$SEGMENT_DIR"/*.pem "$SEGMENT_DIR"/*.crt; do
        [ -e "$cert_file" ] || continue  # Skip if no matching files
        cert_name=$(basename "$cert_file" | sed 's/\.[^.]*$//')  # extentie verwijderen

        echo " - Uploading $cert_name from $cert_file"
        vault kv put "$VAULT_PATH/$cert_name" cert=@"$cert_file"
    done

    # niew manifest maken (lijst van certs) voor fetching op end-points
    echo "manifest maken voor $SEGMENT"
    cert_keys=$(ls "$SEGMENT_DIR" | sed 's/\.[^.]*$//' | jq -R . | jq -s .)
    vault kv put "$VAULT_PATH/_manifest" keys="$cert_keys"
done

echo "Alle certs geupload naar de Vault."
```

Codefragment 4.3: Het bash script dat de root certificaten upload naar de Vault server.

Voor de beveiliging van de root certificaten in de vault zal er gebruik gemaakt worden van token autorisatie. Dit is een manier van autorisatie waarbij een token door de end-points wordt gebruikt om toegang te krijgen tot de vault. Om ervoor te zorgen dat de rechten met deze token gelimiteerd zijn, kan er een policy aangemaakt worden.

De volgende policy kan gebruikt worden voor read-only toegang tot alle root certificaten op alle netwerksegmenten:

```
path "secret/data/certs/*" {
capabilities = ["read", "list"]
}

path "secret/metadata/certs/*" {
capabilities = ["read", "list"]
}

path "secret/data/certs/*/manifest" {
capabilities = ["read"]
}

path "secret/data/certs/*/*" {
capabilities = ["read"]
}
```

Codefragment 4.4: De policy die leesrechten geeft tot de root certificaten in de Vault.

Deze policy kan dan toegevoegd worden aan een role en uiteindelijk kan een token aangemaakt worden met deze role. Eerst moet approle, de manier van authenticatie aangezet worden met het volgende commando:

```
vault auth enable approle
```

Daarna kan de voorgaande policy aangemaakt worden met het volgende commando:

```
vault policy write certs-read public-certs.hcl
```

Waarbij “public-certs.hcl” het bestand is dat de policy bevat en “certs-read” de naam die gegeven wordt aan de policy.

De rol kan dan aangemaakt worden als volgt:

```
vault write auth/approle/role/chef_cert_reader \
token_policies="certs-read" \
secret_id_ttl="60m" \
token_ttl="30m" \
token_max_ttl="60m"
```

Waarbij chef_cert_reader de naam is die gegeven wordt aan de role. Token_policies linkt de policy die zojuist werd gemaakt aan de role, secret_id_ttl is de tijd dat de secret_id geldig is, token_ttl is de tijd dat de token geldig is en token_max_ttl is de maximale tijd dat de token geldig is. Clients kunnen via de role_id en secret_id een token aanvragen, voor het bekomen van de role_id kan er gebruik gemaakt worden van het volgende commando:

```
vault read auth/approle/role/chef_cert_reader
```

Deze role_id blijft geldig tot en met de vault server opnieuw opgestart wordt. De secret_id is ook nodig, deze kan aangemaakt worden met het volgende commando:

```
vault write -f auth/approle/role/chef_cert_reader/secret-id
```

De secret_id heeft een bepaalde levensduur die eerder werd ingesteld bij het maken van de role.

4.4.2. Installeren van een Vault agent

Om de clients toegang te geven tot de vault, moet er een token aangemaakt worden met de role_id en secret_id waarover hiervoor werd gesproken. Om deze token aan te maken, moet er een vault agent draaien op de end-point. Deze agent kan geconfigureerd worden met het volgende bestand '/etc/vault/agent-config.hcl':

```
pid_file = "/run/vault/vault-agent.pid"

vault {
  address = "http://vault.bachelorproef.local:8200"
}

auto_auth {
  method "approle" {
    mount_path = "auth/approle"
    config = {
      role_id_file_path   = "/var/lib/vault/role_id"
      secret_id_file_path = "/var/lib/vault/secret_id"
      remove_secret_id_file_after_reading = false
    }
  }

  sink "file" {
    config = {
      path = "/var/lib/vault/agent-token"
    }
  }
}

cache {
  use_auto_auth_token = true
}

listener "tcp" {
  address     = "0.0.0.0:8201"
  tls_disable = true
}
```

Codefragment 4.5: De gebruikte configuratie voor de Vault agent.

Waarbij “role_id_file_path” het pad is naar het bestand dat de role_id bevat en “secret_id_file_path” het pad is naar het bestand dat de secret_id bevat. Sink “file” zorgt ervoor dat de token die aangemaakt wordt door de agent, opgeslagen wordt in het bestand “/var/lib/vault/agent-token”. Dit bestand kan dan gebruikt worden door de end-point om toegang te krijgen tot de vault.

Om de agent te starten kan er gebruik gemaakt worden van het volgende commando:

```
1    vault agent -config=/etc/vault/agent-config.hcl
```

De agent zal dan automatisch de token aanmaken en hernieuwen wanneer deze verloopt.

Ook op Windows kan Vault agent draaien als een service, helaas door de limitaties van bronnen voor deze proof-of-concept is de installatie hiervan niet uitgevoerd, in de plaats hiervan zal er een token aangemaakt worden in een shared directory op de Windows server.

4.4.3. Oplossing voor Windows end-points met SCCM en Vault

Aangezien het importeren van root certificaten in de GPO's niet geautomatiseerd kan worden, is er geen schaalbaarheid in de vorige oplossing. Dit betekent dat als er veel root certificaten zijn, het handmatig aanmaken van de GPO's een grote kost in tijd zal zijn. Om dit probleem op te lossen kan er in de plaats van root certificaten aan de GPO's te koppelen, een script gemaakt worden dat de root certificaten ophaalt en deze toevoegt aan de truststore van de clients.

Dit kan gedaan worden met het volgende PowerShell script:

```
param (
    [Parameter(Mandatory = $true)]
    [string]$NetworkSegment
)

# Configuration
$VaultAddress = "http://vault.bachelorproef.local:8200"
$LocalCertPath = "C:\Temp\VaultCerts"
$TokenFilePath = "\\bachelor-
 ↪  proef.local\SYSVOL\bachelorproef.local\scripts\archive\token.txt"

# Logging
$LogFile = "C:\Temp\Truststore_Clear_And_Update_Log.txt"
function Write-Log {
    param (
        [string]$Message
    )
    $Timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
    Add-Content -Path $LogFile -Value "$Timestamp - $Message"
}

Write-Log "Script started with NetworkSegment = $NetworkSegment"

# Step 1: Vault token ophalen van bestand
Write-Log "Vault token ophalen op locatie: $TokenFilePath..."
```

Codefragment 4.6: Het Powershell script dat op end-points draait om root certificaten te importeren van de Vault server.

```
try {
    if (-Not (Test-Path $TokenFilePath)) {
        Write-Log "Token file niet gevonden op: $TokenFilePath.
         ↪  Exiting..."
        exit
    }
    $VaultToken = Get-Content -Path $TokenFilePath -ErrorAction Stop
    Write-Log "Vault token succesvol opgehaald"
} catch {
    Write-Log "Kan vault token niet ophalen: $($_.Exception.Message).
     ↪  Exiting..."
    exit
}

# Ensure local cert path exists
New-Item -ItemType Directory -Force -Path $LocalCertPath | Out-Null

# Step 2: Clear Truststore
Write-Log "truststore clearen..."
$Certs = Get-ChildItem -Path Cert:\LocalMachine\Root
foreach ($Cert in $Certs) {
    try {
        Remove-Item -Path $Cert.PSPath -Force
        Write-Log "certificaat verwijderd: $($Cert.Subject)"
    } catch {
        Write-Log "Kon certificaat niet verwijderen:
         ↪  $($Cert.Subject). Error: $($_.Exception.Message)"
    }
}
Write-Log "Truststore cleared."

# Step 3: root certs van vault ophalen

$ManifestUrl =
 ↪  "$VaultAddress/v1/secret/data/certs/$NetworkSegment/_manifest"
try {
    $Manifest = Invoke-RestMethod -Uri $ManifestUrl -Headers @{
     ↪  "X-Vault-Token" = $VaultToken } -Method GET
    Write-Log ("Fetched manifest voor segment $NetworkSegment : " +
     ↪  ($Manifest | ConvertTo-Json -Depth 3))
}

```

Codefragment 4.6: Vervolg op het Powershell script.

```
catch {
    $msg = $_.Exception.Message
    Write-Log "Kon manifest niet fetchen voor '$NetworkSegment': $msg"
    exit
}

# structuur van de manifest controlleren
if ($Manifest.data -and $Manifest.data.data) {
    try {
        $CAList = $Manifest.data.data.keys | ConvertFrom-Json
    }
    catch {
        Write-Log "kon manifest niet parsen naar JSON voor '$NetworkSegment'. Exiting..."
        exit
    }
} else {
    Write-Log "Foutief manifest formaat of geen certificaten gevonden voor '$NetworkSegment'. Exiting..."
    exit
}

if ($CAList.Count -eq 0) {
    Write-Log "Geen certificaten gevonden voor '$NetworkSegment'. Exiting..."
    exit
}

foreach ($CA in $CAList) {
    if ([string]::IsNullOrWhiteSpace($CA)) {
        Write-Log "Lege CA naam gevonden in manifest. overslaan..."
        continue
    }

    $CertUrl = "$VaultAddress/v1/secret/data/certs/$NetworkSegment/$CA"
    Write-Log "Downloading cert: $CA"
```

Codefragment 4.6: Vervolg op het Powershell script.

```
    try {
        $CertData = Invoke-RestMethod -Uri $CertUrl -Headers @{
        ↪ "X-Vault-Token" = $VaultToken } -Method GET
        $CertContent = $CertData.data.data.cert
    }
    catch {
        $msg = $_.Exception.Message
        Write-Log "kon certificaat niet ophalen '$CA': $msg"
        continue
    }

    # als bestand opslaan
    $CertFile = "$LocalCertPath\$NetworkSegment-$CA.cer"
    [System.IO.File]::WriteAllText($CertFile, $CertContent)
    Write-Log "Cert opgeslagen in $CertFile"

    # cert importeren naar truststore
    Write-Log "cert importeren: $CA"
    try {
        Import-Certificate -FilePath $CertFile -CertStoreLocation
        ↪ "Cert:\LocalMachine\Root"
        Write-Log "certificaat geimporteerd"
    }
    catch {
        Write-Log "kon certificaat niet importeren '$CA':
        ↪ $_.Exception.Message"
    }
}

Write-Log "script beeindigd voor netwerksegment $NetworkSegment"

```

Codefragment 4.6: Vervolg op het Powershell script.

Dit script begint met het ophalen van de token voor de Vault server uit een bestand dat op de Windows server staat in de directory \SYSVOL\bachelorproef.local\scripts\archive\token.txt. Dit bestand kan dan gebruikt worden door de end-point om toegang te krijgen tot de vault. Hierna zal de truststore leeggemaakt worden door alle certificaten te verwijderen. Het script haalt dan aan de hand van de parameter “NetworkSegment” de root certificaten op uit de vault en deze worden toegevoegd aan de truststore van de end-points.

Het script logt ook elke stap die uitgevoerd wordt in een log bestand dat opgeslagen wordt in de directory “C:\Temp\Truststore_Clear_And_Update_Log.txt”.

Een belangrijke opmerking bij dit script is dat de hele truststore leeggemaakt wordt, wat in een productieomgeving voor problemen kan zorgen. Dit kan mogelijks opgelost worden door het script idempotent te maken, dit betekent dat het script alleen de root certificaten toevoegt die niet in de truststore staan en alleen de certificaten verwijdert die niet meer in de manifest te vinden zijn. Ook is het belangrijk dat het script op een centraal toegankelijke plaats zit zoals een netwerk drive.

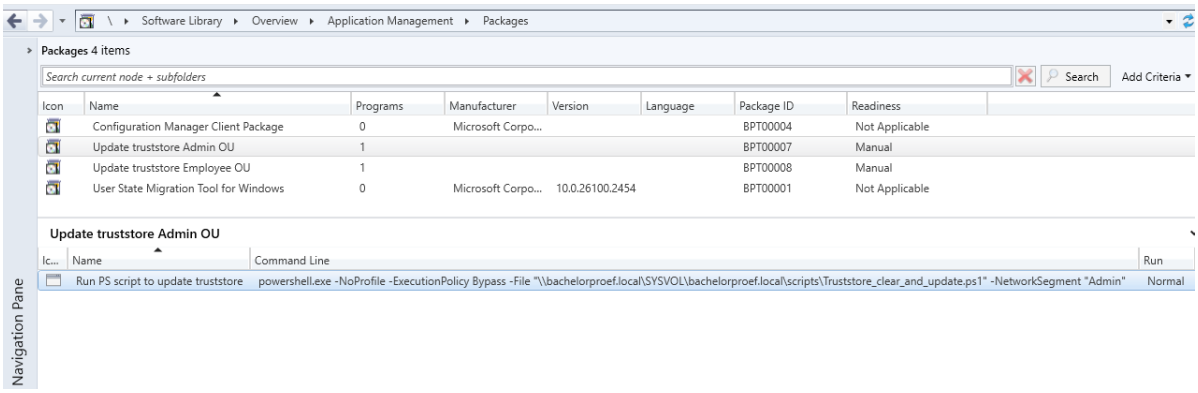
Om ervoor te zorgen dat dit script door de Windows end-points uitgevoerd wordt, kan er gebruik gemaakt worden van SCCM. Hier kunnen we het script als een package toevoegen en deze package kan dan uitgerold worden naar de end-points. Dit kan gedaan worden door een nieuwe applicatie aan te maken in SCCM en het script toe te voegen als een package. Dit kan gedaan worden door de volgende stappen te volgen:

- Open de SCCM console en ga naar “Software Library”.
- Klik op “Applications manager” en dan “Packages”.
- Selecteer “Create package”.

Hier kan dan een naam gegeven worden aan de package zoals “Update truststore Admin OU”, de bron directory kan gezet worden naar de shared folder waarin het script zich bevindt, in dit voorbeeld is dat “\bachelorproef.local\SYSVOL\bachelorproef.local\scripts\”. Kies hierna voor “Standard program” en geef het een naam zoals “Run PS script to update truststore”. Als command line kan het powershell script opgegeven worden met de juiste parameters zoals:

```
1    powershell.exe -ExecutionPolicy Bypass -File "\\bachelor-
2    proef.local\SYSVOL\bachelorproef.local\scripts\update-
3    _and_clear_truststore.ps1" -NetworkSegment "Admin"
```

Zorg ervoor dat de “Run with administrative rights” optie is aangevinkt. Dit zorgt ervoor dat het script met administrator rechten uitgevoerd wordt, wat nodig is alleen om root certificaten te verwijderen uit de truststore. Na het aanmaken van de packages met program zou het er als volgt moeten uitzien:



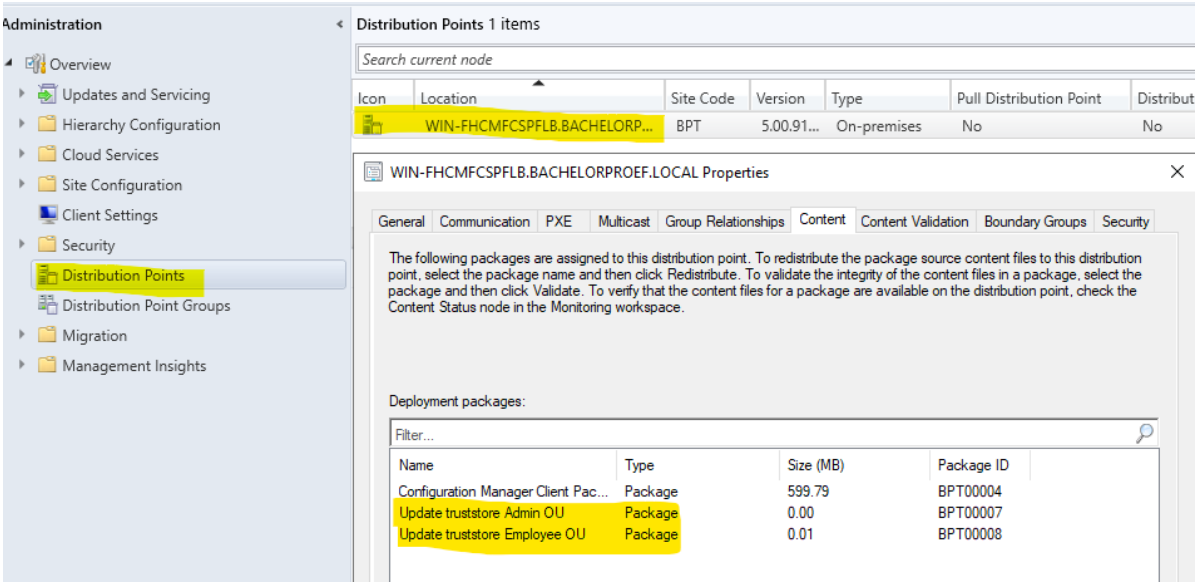
Figuur 4.5: De packages voor de Admin en Employee clients in het SCCM management console.

Na het aanmaken van de package kan deze uitgerold worden naar de end-points door een nieuwe deployment aan te maken. Dit kan gedaan worden door de volgende stappen te volgen:

- Ga naar de SCCM console en ga naar “Software Library”.
- Klik op “Packages” en selecteer de package die je zojuist hebt aangemaakt.
- Klik met de rechtermuisknop op de package en selecteer “Deploy”.
- Selecteer de collectie waar je de package naar wilt uitrollen, in dit geval is dat de “Admin” collectie.
- Selecteer de distributiepunten waar je de package naar wilt uitrollen, in dit geval is dit enkel de Windows Server zelf.
- Zet de “purpose” op “required”.
- Maak een schedule aan voor het uitrollen, dit kan gedaan worden door de “New” optie te selecteren naast “assignment schedule” en dan op “Schedule...” te klikken.
- Selecteer een start datum en bij “configure a reccurance” selecteer “custom interval” en bepaal hoe frequent de truststore moet geüpdatet worden.
- Bij “user experience” selecteer “Software installation” en vink de optie “commit changes at deadline” af.
- Als de deployment optie kies “Download content from distribution point and run locally”.

Hierna kan de deployment gefinaliseerd worden. De bovenstaande stappen kunnen dan herhaald worden voor de andere netwerksegmenten.

Na het deployen van de packages kan je controlleren of de distribution succesvol was door naar “Administration” te gaan en dan naar “Distribution points” en bij properties van de distribution point te kijken of de packages bij content te vinden zijn zoals in de volgende afbeelding:



Figuur 4.6: De deployed packages in de distribution point content.

End-points die binnen de groep van de deployment zaten kunnen de deployment controlleren door naar “Software Center” te gaan en daar naar de package te zoeken. De package zal vermelden wanneer het ingepland is om uitgevoerd te worden.

4.4.4. Oplossing voor Linux end-points met Chef en Vault

Een probleem met de vorige oplossing is dat Ansible een lage schaalbaarheid heeft. Dit betekent dat als er veel end-points zijn, de Ansible playbook traag zal zijn en mogelijk niet alle end-points kan bereiken.

Om dit probleem aan te pakken kan er gebruikt gemaakt worden van Chef. Deze tool verschilt van Ansible op het vlak van wie de instructies uitvoert. Bij Ansible is de Ansible control server verantwoordelijk voor het uitvoeren van de instructies op de end-points, terwijl bij Chef de end-points zelf verantwoordelijk zijn voor het uitvoeren van de instructies. Hiervoor is er beide nood aan een Chef infra server en een Chef infra client. De infra server is de server die de instructies zal geven aan de end-points en de infra client is de end-point die de instructies zal uitvoeren op de end-points.

De infra server kan geconfigureerd worden met het volgende commando:

```
1    chef-server-ctl reconfigure
```

Dit zal prompten om de naam van de server, de organisatie en de naam van de validator. Dit zijn de gegevens die gebruikt worden om de infra server te configureren. Voor deze proof-of-concept zal de infra server draaien op de server “CA” en zullen de naam, organisatie en validator respectievelijk “ca”, “bachproef” en “bachproef-validator” zijn.

Chef maakt gebruikt van cookbooks, dit zijn bestanden die de instructies bevatten voor de end-points. Om cookbooks te maken en te uploaden naar de chef infra server, moet er een chef workstation geïnstalleerd worden. Binnen deze omgeving bevindt deze zich op de server “Database” aangezien deze server momenteel geen andere taken heeft.

Chef workstation kan gebruikt worden met het “knife” commando, dit is een command line tool die gebruikt wordt om te communiceren met de chef infra server. Chef workstation wordt geconfigureerd met het volgende bestand ’knife.rb’:

```
current_dir = File.dirname(__FILE__)
log_level                 :info
log_location              STDOUT
node_name                 'db.bachelorproef.local'
client_key                "/home/almalinux/.chef/bpchef.pem"
chef_server_url
  ↪  'https://ca.bachelorproef.local/organizations/bachproef'
cookbook_path             ["/home/almalinux/cookbooks/"]
ssl_verify_mode :verify_none
editor 'vi'
```

Codefragment 4.7: Het gebruikte configuratiebestand voor de Chef workstation.

ssl_verify_mode is ingesteld op "verify_none" aangezien dit een testomgeving is. In een productieomgeving zou dit ingesteld moeten worden op "verify_peer" waarbij het certificaat van de chef infra server gevalideerd wordt.

Eenmaal chef workstation is geconfigureerd kunnen er cookbooks gemaakt worden. Zoals in het configuratiebestand te zien is, worden de cookbooks opgeslagen in de directory "/home/almalinux/cookbooks/". Voor het oplossen van het probleem in dit onderzoek zal er een cookbook gemaakt worden dat de root certificaten uit de vault haalt en deze toevoegt aan de truststore van de end-points.

Elke cookbook moet zijn eigen subdirectory hebben binnen de cookbooks directory. In dit geval werd de cookbook 'truststore_update' aangemaakt in de directory "/home/almalinux/cookbooks/truststore_update/". Elke cookbook moet een metadata.rb bestand hebben dat de naam van de cookbook bevat. Dit bestand ziet er als volgt uit:

```
    cookbooks/truststore_update/metadata.rb
```

Naast de metadata.rb moet er een recipe directory zijn die de instructies van de cookbook bevat en kan eventueel een attributes directory gemaakt worden die default waarden voor attributen gedefinieerd heeft.

De cookbook die voor deze oplossing zal gebruikt worden is te vinden in de directory "/home/almalinux/cookbooks/truststore_update/recipes/default.rb" en ziet er als volgt uit:

```
require 'vault'
require 'json'
require 'fileutils'

Vault.address = 'http://vault.bachelorproef.local:8200'
Vault.token = ::File.read("/var/lib/vault/agent-token").strip

network_segment = node['network_segment'] || raise("Missing
 ↪  network_segment attribute")

# manifest ophalen
Chef::Log.info("manifest ophalen voor netwerk segment:
 ↪  #{network_segment}")
manifest_secret =
 ↪  Vault.logical.read("secret/data/certs/#{network_segment}/_manifest")
manifest_data = manifest_secret.data[:data]

Chef::Log.info("Manifest data: #{manifest_data.inspect}")

# 'keys' veld ophalen uit manifest_data
keys = manifest_data[:keys]

# debug
Chef::Log.info("Raw 'keys' value: #{keys.inspect}")

# 'keys' verwerken naar JSON'
cert_list = []
begin
  Chef::Log.info("'keys' veld naar JSON parsen...")
  cert_list = JSON.parse(keys.strip)
rescue JSON::ParserError => e
  Chef::Log.warn("Kon keys niet naar JSON parsen: #{e.message}.
   ↪  Vermoedelijk is de lijst leeg.")
end

# parsed cert_list loggen voor debugging
Chef::Log.info("Parsed cert_list: #{cert_list.inspect}")

# target directory bepalen op basis van distributie

```

Codefragment 4.8: Het Chef cookbook dat de root certificaten ophaalt van de Vault server en importeert.

```
target_directory = case node['platform_family']
                when 'debian'
                    "/usr/local/share/ca-certificates/"
                when 'rhel'
                    "/etc/pki/ca-trust/source/anchors/"
                else
                    raise "Unsupported platform family"
                end

# Bestaande certificaten uit directory verwijderen
Chef::Log.info("Huidige certificaten verwijderen uit #{target_directory}")
Dir.glob("#{target_directory}*.crt").each do |file|
  Chef::Log.info(" - Deleting file: #{file}")
  FileUtils.rm(file)
end

# Als cert_list leeg is, warning loggen en doorgaan
if cert_list.empty?
  Chef::Log.warn("geen certificaten in manifest. Trust store wordt geupdate met lege directory.")
else
  # Elk certificaat ophalen en opslaan
  cert_list.each do |cert_name|
    Chef::Log.info("certificaat verwerken: #{cert_name}")
    cert_path = "secret/data/certs/#{network_segment}/#{cert_name}"
    Chef::Log.info("Certificaat van volgend pad ophalen: #{cert_path}")

    cert_secret = Vault.logical.read(cert_path)
    Chef::Log.info("Vault antwoord voor #{cert_path}: #{cert_secret.inspect}")

    cert_pem = cert_secret.data[:data][:cert]
    Chef::Log.info("Certificaat content voor #{cert_name}: #{cert_pem.inspect}")

    # zorgen dat cert_pem niet nil of empty is
    if cert_pem.nil? || cert_pem.strip.empty?
      raise "Certificaat content voor #{cert_name} is leeg"
    end

    target_path = "#{target_directory}#{cert_name}.crt"

```

Codefragment 4.8: Vervolg van het cookbook.

```
    file target_path do
      content cert_pem
      owner 'root'
      group 'root'
      mode '0644'
    end
  end
end

# Update truststore
Chef::Log.info("truststore updaten...")
execute 'update-trust-store' do
  command case node['platform_family']
          when 'debian'
              'update-ca-certificates'
          when 'rhel'
              'update-ca-trust extract'
          else
              raise "Unsupported platform family"
          end
  action :run
end

```

Codefragment 4.8: Vervolg van het cookbook.

De directory attributes bevat een default.rb bestand dat er als volgt uitziet:

```
    default['platform_family'] = 'rhel'
```

De reden voor deze default waarde is omdat de RHEL image die gebruikt werd incorrect de platform_family teruggeeft.

Met alle nodige bestanden en directories kan de cookbook worden geüpload naar de chef infra server met het volgende commando:

```
    knife cookbook upload truststore_update
```

End-points hebben dan een chef client nodig om de cookbooks te kunnen opvragen en uitvoeren. Deze kan geïnstalleerd worden vanuit de chef workstation met het volgende commando:

```
    knife bootstrap <ip_address> \
  --ssh-user <user> \
  --sudo \
  --use-sudo-password \
  --node-name <node_name> \
  --run-list 'recipe[truststore_update]' \
  --json-attributes '{"network_segment": "segment_name"}'
```

Dit commando zou er als volgt uitzien voor de server “webserver” met hostname “bpweb”:

```
    knife bootstrap bpweb.bachelorproef.local \
  --ssh-user ubuntu \
  --sudo \
  --use-sudo-password \
  --node-name bpweb \
  --run-list 'recipe[truststore_update]' \
  --json-attributes '{"network_segment": "DMZ"}'
```

Elke Chef client moet dus een attribuut “network_segment” hebben, deze variabele wordt gebruikt voor de juiste subdirectory te kiezen in de vault. Daarnaast de run-list parameter ingevuld worden met de naam van de cookbook die eerder werd gemaakt.

Nu moeten de clients het cookbook uitvoeren, dit kan gedaan worden met het volgende commando:

```
    chef-client
```

De client zal alle cookbooks ophalen van de chef infra server en deze uitvoeren. Dit kan ook geconfigureerd worden dat dit automatisch gebeurd op een bepaalde tijd, dit kan gedaan worden met het volgende commando:

```
    echo "chef-client -i 1800" >> /etc/cron.d/chef-client
```

Dit zorgt ervoor dat de chef-client elke 30 minuten zal draaien en de cookbooks zal ophalen van de chef infra server. Met deze opstelling zouden de Linux end-points nu steeds up-to-date moeten zijn met de root certificaten in de vault.

4.4.5. Pro's en Con's van de tweede oplossing

- Pro's:
  - Root certificaten kunnen beheerd worden op een centrale plaats.
  - Met de SCCM task sequence blijven de Windows end-points up-to-date ook als ze niet opnieuw opgestart worden.
  - Alle workload wordt gedaan door de end-points zelf, wat zorgt voor hogere schaalbaarheid.
- Con's:
  - De oplossing is afhankelijk van verschillende servers, wat voor meer points of failure zorgt.
  - Er zijn meer servers en software die onderhoud nodig hebben.

4.5. Overwegingen en aanbevelingen

Deze proof-of-concept is een goede start voor het aantonen van de mogelijkheden tot het centraal beheren van root certificaten op een netwerk. Er zijn echter nog enkele overwegingen en aanbevelingen die gedaan kunnen worden om de oplossing te verbeteren.

- Alhoewel deze proof-of-concept de trust verdeelt op niveau van netwerksegmentatie, kan deze laag gelegd worden waar nodig, zolang de Linux end-points de juiste Chef attributen hebben of in de juiste groep binnen de Ansible inventory file zitten en de Windows end-points in de juiste OU's en SCCM groepen zitten.
- Certificaten die door alle end-points gebruikt worden, kunnen mogelijks in een aparte directory staan in de vault. Zo moeten deze certificaten niet aan elke directory van elk netwerksegment toegevoegd worden. Hiervoor moeten dan ook de scripts en instructies aangepast worden.
- De huidige scripts en instructies kunnen verder geoptimaliseerd worden door een controle te implementeren die kijkt naar welke certificaten al aanwezig zijn in de truststore om ze te vermijden dat er onnodige transfers en downtime plaatsvinden.
- Momenteel zijn de enige applicatie specifieke truststores die beïnvloed zijn, diegene die de RHEL systeemtruststore ondersteund, andere applicatietruststores, of applicatietruststores op andere Linux distributies zijn niet opgenomen in deze proof-of-concept. Dit kan verder onderzocht worden.
- Naast Linux en Windows end-points hebben bedrijven vaak ook MacOS end-points en IoT devices. De ondersteuning van deze devices zou ook verder onderzocht moeten worden.

- In de realiteit is het misschien beter om de updates van de truststores af te stemmen op acties van een bestaande PKI in de plaats van vaste tijdsintervallen. Hier zal dan moeten gekeken worden hoe dit geïmplementeerd kan worden.
- De verschillende servers zoals SCCM, AD, Chef, Ansible en Vault kunnen in de verschillende netwerksegmenten opgezet worden en meerder maal aanwezig zijn om availability te garanderen.

5

Conclusie

Het onderzoek naar een centraal trust management systeem was succesvol en heeft geleid tot meerdere mogelijkheden om centraal de truststores van alle Windows en Linux systemen in een netwerk te beheren, waarbij ervoor gezorgd kan worden dat niet elk systeem dezelfde root certificaten heeft.

Hiervoor kan een centrale opslagplaats zoals een Hashicorp Vault gebruikt worden om de root certificaten op te slaan en beheren, waarbij er een indeling wordt gemaakt per netwerksegment met daarin alleen de root certificaten die nodig zijn voor dat segment.

De Windows end-points kunnen deze certificaten gemakkelijk toegewezen krijgen door gebruik van Group Policy Objects (GPO's), maar Microsoft biedt geen mogelijkheid aan in Powershell om certificaten toe te voegen of verwijderen uit de GPO's wat het een lage schaalbaarheid geeft op vlak van het aantal root certificaten dat kan toegevoegd worden. Als alternatief kan er gebruik gemaakt worden van System Center Configuration Manager (SCCM) samen met een Powershell script dat de certificaten voor het juiste netwerksegment uit de Vault haalt. Het script kan uitgerold worden als een package voor de Windows end-points om zo ingepland worden om op regelmatige basis de root certificaten te vernieuwen.

Aan de Linux kant kan er gebruik gemaakt worden van zowel Ansible als Chef om zo instructies uit te voeren op de Linux end-points die net zoals het Powershell script de root certificaten uit de Vault haalt en deze toevoegt aan de truststore van het systeem. Ansible kan meer tijd in beslag nemen om de root certificaten te vernieuwen op een groot aantal end-points, maar heeft een makkelijker opzet en is gebruiksvriendelijker dan Chef. Maar Chef legt de verantwoordelijkheid van het uitvoeren van de instructies bij de end-points zelf, wat het een snellere oplossing

maakt voor het vernieuwen van de root certificaten ook bij een groter aantal end-points.

De configuratie van deze oplossingen werden geïmplenteerd in een proof-of-concept omgeving die een basis kan leggen voor bedrijven om te bepalen welke oplossing het beste past bij hun infrastructuur en voorkeuren. Uit de oplossing bleek dat ondanks hier de verschillende indelingen van de vertrouwde root certificaten gebaseerd zijn op de netwerksegmenten, dat de indeling gemaakt kan worden op eender welk criterium dat het bedrijf wenst, zolang de SCCM groepen, OU's, Ansible inventory of Chef nodes de correcte configuratie hebben.

Bedrijven kunnen dan zelf verder onderzoeken hoe deze oplossing op een grotere schaal kan gebruikt worden om availability en security te garanderen. Daarnaast kan er ook verder onderzocht worden hoe bedrijven de updates van de truststores kunnen afstemmen met functionaliteiten van hun bestaande PKI's.

A

Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1. Inleiding

In de digitale wereld is vertrouwen in beveiligingssystemen cruciaal, vooral binnen de financiële sector. Banken hebben complexe netwerken die moeten worden beschermd tegen cyberbedreigingen. Trust management, het beheren van vertrouwde relaties aan de hand van digitale certificaten en public key infrastructures (PKI's) speelt hierin een belangrijke rol.
Daarbij is het effectief beheren van truststores en de certificaten die erin worden vertrouwd, de verzameling van vertrouwde certificaten, essentieel om veilige gegevensuitwisseling en transacties te garanderen.
In vele netwerken worden truststores verspreid beheerd, hun locatie is afhankelijk van de gebruikte applicatie of besturingssystemen en de bedrijfsrollen. Dit leidt tot een gefragmenteerde aanpak, wat niet alleen inefficiënt is, maar ook de kans vergroot op verouderde certificaten en beveiligszwaktes.
Door een centrale aanpak te ontwikkelen voor trust management wordt er gestreefd naar meer consistentie, eenvoud en veiligheid in het beheer van certificaten. Dit onderzoek richt zich op het formuleren van richtlijnen en oplossingen die specifiek van toepassing op de behoeftes van de financiële sector.
Om dit doel te bereiken, beantwoordt dit onderzoek de volgende deelvragen:
1. Wat zijn de belangrijkste concepten en technologieën achter PKI en trust managment?
2. Welke tools en technieken worden momenteel gebruikt voor trust management, en wat zijn hun beperkingen?

3. Welke specifieke uitdagingen en eisen gelden voor trust management in bankomgevingen?
4. Welke componenten moeten worden opgenomen in een virtueel bankennetwerk om een realistisch beeld te creëren?
5. Hoe kan een gecentraliseerde oplossing voor trust management worden ontworpen en geïmplementeerd in een bankennetwerk?
6. Hoe kan de veiligheid, efficiëntie en schaalbaarheid van de gecentraliseerde oplossing worden getest en geëvalueerd?
7. In hoeverre voldoet de gecentraliseerde oplossing aan de beveiligings- en complianceeisen van een bankomgeving?
8. Wat zijn de praktische voordelen en beperkingen van een gecentraliseerde trust management oplossing?
9. Welke aanbevelingen kunnen worden gedaan voor banken die een centrale trust management systeem willen implementeren?

Door deze vragen te beantwoorden, streeft het onderzoek ernaar richtlijnen en oplossingen te formuleren die aansluiten bij de specifieke behoeften van de financiële sector.

A.2. Literatuurstudie

Bij het begrijpen van trust management is het belangrijk om te weten wat een public key infrastructure (PKI) en digitale certificaten zijn.

Een digitaal certificaat is een file die verbonden is met een cryptografische key pair en het authenticeert de identiteit van een website, individu, organisatie, gebruiker of apparaat. Zulke certificaten worden soms ook public key certificates of identity certificates genoemd

Een certificaat bevat het onderwerp, wat dient als de identiteit samen met een digitale signatuur. Het doel van digitale certificaten is om de identiteit en encryptie te verzekeren van een website, individu, organisatie, apparaat, gebruiker of server. (Digicert, 2025)

Een PKI beveiligd data en bouwt vertrouwen via deze digitale certificaten, die uitgegeven worden door certificate autorities (CA's) aan bedrijven nadat zij een certificate signing request (CSR) hebben gemaakt. Eens een CSR is goedgekeurd en een certificaat is uitgegeven kan een bedrijf of individu zijn communicatie, web domeinen en/of documenten authenticeren en beveiligen zolang het certificaat is gemaakt voor dit doeleinde en geldig is. (Topping, 2023)

Naast PKI en digitale certificaten zijn truststores en keystores ook van belang bij trust management.

IBM (2023) definieert dat Truststores en keystores cryptografische artefacten bevatten, met andere woorden certificaten en private keys. Deze artefacten worden dan gebruikt door protocollen zoals TLS. Een keystore bevat persoonlijke certificaten samen met de overeenkomstige private keys die gebruikt worden om de eigenaar

van het certificaat te identificeren.
Voor een TLS verbinding stelt een persoonlijk certificaat de identiteit van een endpoint voor, beide de client en de server hebben dit certificaat om elkaar te identificeren.
Een truststore bevat dan weer signer certificates die door de andere endpoints vertrouwd worden, deze certificates zijn ook gekend als ”certificate autority certificates”.
Deze signer certificate bevatten een public key die gebruikt wordt om een persoonlijk certificaat te valideren. Door het toevoegen van de signer certificate in de client zijn truststore, kan de client de server vertrouwen om een TLS verbinding te maken.
Vele software en besturingssystemen komen dan ook vooraf geïnstalleerd met hun eigen truststores.
”Er zijn 4 grote organisaties die zulke truststores beheren: 1. Microsoft root certificate program die gebruikt wordt voor Windows. 2. Apple root certificate program is gebruikt voor alle Mac apparaten. 3. De Mozilla root certificates program wordt gebruikt door Mozilla zelf en meeste Linux distributies. 4. Google root certificate program wordt gebruikt door Google chrome en andere Google applicaties.”(Dwivedi, 2024)
Volgens Mervana (2024) is een van de uitdagingen voor het beheren van truststores, dat in grote organisaties met heterogene netwerken vele truststores gedecentraliseerd liggen waarbij er een nood is aan gecoördineerd management.
Dit kan opgelost worden door een gecentraliseerd management systeem, dit maakt het makkelijker om veranderingen te coördineren.
Andere bronnen raden zulke gecentraliseerde systemen sterk aan.
”Organisaties moeten overwegen de default truststores af te wijzen. Ze zouden een eigen corporate-level truststore moeten maken aan de hand van certificate whitelisting om te bepalen welke certificaten hierin worden opgenomen.”(Arampatzis, 2020)
Bij het opstellen van deze centrale truststore binnen de financiële sector brengt dit ook enkele uitdagingen met zich mee. Er zijn namelijk beveiligingseisen voor de infrastructuur van banken waaraan moet worden voldaan.
Daarbij blijkt dat tegen het einde van 2024 de cyber security binnen europese banken moeten voldoen aan de ”Digital Operational Resilience Requirements”(DORA). (Gorobeț, 2024)
Naast de DORA eisen zijn er ook nog andere eisen die van toepassing zijn op de financiële sector zoals de ”Payment Services Directive 2”(PSD2).
In het onderzoek van Gounari e.a. (2024) blijkt dat PSD2 de nieuwe vereisten binnen communicatie systemen baseert op SCA, de nieuwe requirement die specifieke technische standaarden introduceert zoals PSD2-compliant certificaten.
In het uitvoeren van dit onderzoek zal er dus moeten worden gekeken naar welke richtlijnen en eisen van toepassing zijn voor het gecentraliseerde management sys-

teem alsook naar hoe deze kunnen worden geïmplementeerd.
Je mag deze sectie nog verder onderverdelen in subsecties als dit de structuur van de tekst kan verduidelijken.

A.3. Methodologie

Het onderzoek naar centrale trust management in een banknetwerk zal opgedeeld worden in drie fases: een literatuurstudie, praktijkstudie en uiteindelijke rapportage en oplevering.
In de eerste maand zal de eerste fase van het onderzoek uitgevoerd worden, de literatuurstudie.
Het doel in deze fase is om een diepgaand begrip te krijgen over trust management, certificaatbeheer en de verschillende aanpakken bij trust managment.
In deze literatuurstudie zal er gekeken worden naar de bestaande theorieën, technologieën en andere tools die momenteel worden gebruikt in praktijk.
Ook zal er gekeken worden naar de voordelen van gecentraliseerde trust management en de eisen binnen de financiële sector.
Op het einde van de eerste fase zal er samen gezeten worden met de co-promotor om mogelijke oplossingen te bespreken die verder zullen worden uitgewerkt in de praktijkstudie.
De info die verkregen wordt in deze literatuurstudie zal de basis leggen voor de volgende fase van dit onderzoek: de praktijkstudie.
Deze fase zou starten in maart en loopt tot eind april met een duur van 8 weken.
Deze praktijkstudie opgedeeld in 3 delen.
het ontwerpen van een virtuele omgeving, de implementatie van het centraal trust management systeem en de evaluatie van de oplossing.
In de eerste stap zal er weer worden samengezeten met de co-promotor om te bepalen welke componenten van het banknetwerk gesimuleerd moeten worden in de virtuele testomgeving.
Hierbij wordt gekeken naar de typische netwerkarchitectuur van een bank, inclusief servers (zoals applicatieservers en webservers), firewalls, proxyservers en verschillende besturingssystemen (zoals Linux en Windows).
Ook wordt er een certificate authority (CA) opgezet om het beheer van certificaten te simuleren.
Deze virtuele omgeving zal worden opgezet binnen GNS3, een netwerkemulator die het mogelijk maakt om complexe netwerken te simuleren.
Het doel is om een realistisch netwerk te creëren dat de complexiteit van een bankomgeving weerspiegelt.
De tweede stap, de implementatie van centraal trust management, begint midden maart en duurt drie weken.
In deze fase wordt een gecentraliseerde oplossing voor trust management opgezet. Dit gebeurt door middel van de tools die gekozen werden samen met de co-

promotor in de literatuurstudie in combinatie met zelfgeschreven scripts voor het certificaatbeheer.
Het doel hier is dat bij het genereren of updaten van een nieuw root certificaat, de truststores op elk systeem in het netwerk consistent zijn met de truststores van de systemen met dezelfde bedrijfsrollen.
Na de implementatie volgt de derde stap van de praktijkstudie, die bestaat uit de evaluatie en validatie van de oplossing.
Deze fase begint begin april en duurt drie weken.
De centrale vraag is hoever de geïmplementeerde oplossing voldoet aan de beveiligingseisen (zoals DORA, PSD2, GDPR, ISO,...) van een bankennetwerk.
Er worden testcases opgesteld om veelvoorkomende uitdagingen te simuleren, zoals het herroepen van een rootcertificaat of het omgaan met verlopen certificaten.
Ook wordt de schaalbaarheid en efficiëntie van de oplossing getest.
Bij de schaalbaarheid zal er gekeken worden naar hoe de oplossing omgaat met het veranderen of uitbreiden van het netwerk en/of het aantal certificaten. Bij de efficiëntie zal er gekeken worden naar hoe snel aanpassingen van certificaten binnen de centrale truststore verspreid worden binnen het netwerk zonder het verstoren van andere diensten.
Als laatste fase van dit onderzoek zal er een rapportage gemaakt worden van de configuraties binnen de POC omgeving. Dit zal gebeuren in het begin van mei met een tijdsduur van 4 weken. De rapportage zal de ontwerpkeuzes en oplossing beschrijven alsook de evaluatie en resultaten van deze oplossing.
Op basis van de bevindingen worden dan conclusies en aanbevelingen geformuleerd voor de implementatie van centraal truststorebeheer in banknetwerken.

A.4. Verwacht resultaat, conclusie

Dit onderzoek verwacht dat gecentraliseerde trust management verbeteringen oplevert in beveiliging, efficiëntie en schaalbaarheid voor een banknetwerk.
Door de certificaten op een centrale plaats te beheren, wordt het risico op verlopen of ongeldige certificaten verkleind, wat de volledige netwerkbeveiliging versterkt.
Ook kan er makkelijker worden aangepast welke apparaten welke certificaten vertrouwen, zo kunnen onnodige certificaten gemakkelijk worden verwijderd uit een truststore.
Het gecentraliseerde beheer zal ook de efficiëntie verhogen door automatisering van certificaatdistributie, wat tijd bespaart en menselijke fouten vermindert.
Bovendien maakt de oplossing schaalbaarheid mogelijk, wat essentieel is voor de groei van banknetwerken.
Dit onderzoek zal praktische aanbevelingen bieden voor de implementatie van deze oplossing in bankomgevingen, waardoor banken hun certificaatbeheer kun-

nen optimaliseren en hun netwerkbeveiliging kunnen versterken.
In conclusie verwacht dit onderzoek te bevestigen dat gecentraliseerd trust management niet alleen de veiligheid verbetert, maar ook bijdraagt aan een efficiënter en beter beheersbaar netwerk, met duidelijke voordelen voor de banksector.

Bibliografie

Arampatzis, A. (2020). What Is a Trust Store and How Hard Is It to Manage? *TLS certificates*. Verkregen februari 26, 2025, van https://venafi.com/blog/what-trust-store-and-how-hard-it-manage/

Canonical Ltd. (2025). Install a root CA certificate in the trust store. *Ubuntu Server documentation*. Verkregen maart 6, 2025, van https://documentation.ubuntu.com/server/how-to/security/install-a-root-ca-certificate-in-the-trust-store/index.html

Digicert. (2025). What is a digitalcertificate? *Trust PKI*. https://www.digicert.com/faq/trust-and-pki/what-is-a-digital-certificate-and-why-are-digital-certificates-important

Dwivedi, D. (2024). What is a Trust Store and the Issues Associated with It. *Encryption Consulting*. https://www.encryptionconsulting.com/what-is-a-trust-store-and-the-issues-associated-with-it/

Goel, A. (2024). What is Certificate Enrollment and how is it used? Verkregen februari 26, 2025, van https://www.encryptionconsulting.com/education-center/what-is-certificate-enrollment-and-how-is-it-used/

Gorobeț, I. (2024). Convergence of banking cybersecurity strategies to the new rules on digital operational resilience. *Proceedings of International Conference "Economic Security in the Context of Systemic Transformations"*. https://doi.org/10.53486/escst2023.06

Gounari, M., Stergiopoulos, G., Pipyros, K., & Gritzalis, D. (2024). Harmonizing open banking in the European Union: an analysis of PSD2 compliance and interrelation with cybersecurity frameworks and standards. *International Cybersecurity Law Review*, *5*(1), 79–120. https://doi.org/10.1365/s43439-023-00108-8

Hashicorp, Inc. (z.d.). What is Vault? *Documentation*. Verkregen mei 4, 2025, van https://developer.hashicorp.com/vault/docs/about-vault/what-is-vault

IBM. (2023, oktober). IBM z/OS Connect. https://www.ibm.com/docs/en/zos-connect/zosconnect/3.0?topic=connect-keystores-truststores

Mervana, P. (2024). What is a Trust Store and How to Manage It? *SSL Insight*. https://sslinsights.com/what-is-trust-store-and-how-to-manage-it/

Microsoft. (2024a). Trusted Root Certification Authorities Certificate Store. *Microsoft Learn*. Verkregen maart 6, 2025, van https://learn.microsoft.com/en-us/windows-hardware/drivers/install/trusted-root-certification-authorities-certificate-store

Microsoft. (2024b). What is Configuration Manager? *Microsoft Learn*. Verkregen maart 12, 2025, van https://learn.microsoft.com/en-us/mem/configmgr/core/understand/introduction

Microsoft. (2025). certutil. *Microsoft Learn*. Verkregen maart 6, 2025, van https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/certutil

Mozilla. (z.d.). Network Security Services (NSS). *Firefox Source Docs*. Verkregen maart 6, 2025, van https://firefox-source-docs.mozilla.org/security/nss/index.html

Mozilla. (2018). NSS Shared DB. *Mozilla wiki*. Verkregen maart 6, 2025, van https://wiki.mozilla.org/NSS_Shared_DB

Mozilla. (2024). Set up Certificate Authorities (CAs) in Firefox. *Support Mozilla*. Verkregen maart 6, 2025, van https://support.mozilla.org/en-US/kb/setting-certificate-authorities-firefox

Netcraft LTD. (2025). Web Server Survey. *January 2025 Web Server Survey*. Verkregen maart 12, 2025, van https://www.netcraft.com/blog/january-2025-web-server-survey/

Okta, Inc. (2023). What Is Certificate Verification for Root CA, Intermediate CA, and End-Entity CA. *Knowledge base*. Verkregen februari 26, 2025, van https://support.okta.com/help/s/article/How-can-we-do-Certificate-Verification-Need-more-explanation-for-Root-CA-Intermediate-CA-End-entity-CA?language=en_US

Perl, H., Fahl, S., & Smith, M. (2014). You Won't Be Needing These Any More: On Removing Unused Certificates from Trust Stores (N. Christin & R. Safavi-Naini, Red.). *Financial Cryptography and Data Security*, 307–315. https://doi.org/https://doi.org/10.1007/978-3-662-45472-5_20

Progress Software Corporation. (2024). Platform Overview. *Progress Chef Docs*. Verkregen maart 7, 2025, van https://docs.chef.io/platform_overview/

Red Hat, Inc. (z.d.-a). Ansible Collaborative. *Ansible*. Verkregen maart 6, 2025, van https://www.redhat.com/en/ansible-collaborative

Red Hat, Inc. (z.d.-b). Chapter 2. Introduction to Puppet. *OpenShift Container Platform*. Verkregen maart 6, 2025, van https://docs.redhat.com/en/documentation/openshift_container_platform/2/html/puppet_deployment_guide/chap-Introduction_to_Puppet#chap-Introduction_to_Puppet

Red Hat, Inc. (z.d.-c). How Ansible works. *Ansible*. Verkregen maart 6, 2025, van https://www.redhat.com/en/ansible-collaborative/how-ansible-works

Red Hat, Inc. (2022). 4.14. Using Shared System Certificates. *Red Hat Documentation*. Verkregen maart 6, 2025, van https://www.redhat.com/en/blog/configure-ca-trust-list

Reddy, R., & Wallace, C. (2010, oktober 1). Trust Anchor Management Requirements. https://doi.org/10.17487/RFC6024

SSL Support Team. (2024). What is the Role of a Certificate Authority? *What is a Certificate Authority (CA)?* Verkregen februari 25, 2025, van https://www.ssl.com/article/what-is-a-certificate-authority-ca/

Stack Exchange Inc. (2024). Technology. *2024 Developer Survey*. Verkregen maart 12, 2025, van https://survey.stackoverflow.co/2024/technology

Thales. (z.d.). What is PKI? *What is PKI and What is it used for?* Verkregen februari 26, 2025, van https://cpl.thalesgroup.com/faq/public-key-infrastructure-pki/what-public-key-infrastructure-pki

Topping, S. (2023). Understanding Public Key Infrastructure: Overview and Key Concepts. *GlobalSign Blog*. https://www.globalsign.com/en/blog/understanding-pki-overview-and-key-concepts

VMware, Inc. (2025). about Salt. *about Salt Project*. Verkregen maart 6, 2025, van https://docs.saltproject.io/en/latest/topics/about_salt_project.html#about-salt