HO GENT

Faculteit Bedrijf en Organisatie

Secrets management: centraal beheer van gevoelige data

Rayen Nasra

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

Promotor:
Antonia Pierreux
Co-promotor:
Jan Delamper

Instelling: HoGent

Academiejaar: 2020-2021

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Secrets management: centraal beheer van gevoelige data

Rayen Nasra

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

Promotor:
Antonia Pierreux
Co-promotor:
Jan Delamper

Instelling: HoGent

Academiejaar: 2020-2021

Tweede examenperiode

Woord vooraf

Deze thesis “Secrets management: centraal beheer van gevoelige data“ werd geschreven in samenwerking met het development team van Wolters Kluwer Financial Services. Deze thesis dient als informatieve bron & als handleiding om de stappen te reproduceren voor te behalen wat met deze proef verwezenlijkt werd. Ik heb dit onderwerp gekozen omdat ik wist dat de resultaten potentieel zouden helpen bij het beheer van gevoelige data doorheen het IT-ecosysteem van Wolters Kluwer.

Deze proef vormt dan ook een hoogtepunt tijdens mijn laatste stappen om mijn studies van de opleiding “Toegepaste Informatica met afstudeerrichting Systeem en Netwerkbeheer“, te voltooien. Het gekozen onderwerp was zeer interessant om me hierin te verdiepen en kwam met veel uitdagingen die getackeld moesten worden. Dit alles zou ik nooit volbracht hebben zonder de hulp van een aantal personen en hiervoor neem ik graag de tijd om deze mensen te bedanken.

Allereerst wil ik mijn promotor, Antonia Pierreux, bedanken voor alle hulp die ze mij heeft aangeboden door middel van meerdere feedback momenten en mijn teksten meerdere keren door te nemen waarmee ik aanpassingen kon verrichten waar nodig. Ze heeft mij ook sterk gesteund om de juiste beslissingen op tijd te nemen.

Ten tweede wil ik ook graag mijn co-promotor, Jan Delamper & Herman van der Merwe bedanken voor alle hulp tijdens mijn stage bij Wolters Kluwer & de mogelijkheid om de proof of concept uit te werken via middelen van het bedrijf.

Als laatste wou ik zeker ook mijn vrienden en medestudenten, Owen van Damme, Emiel van Belle, Olivier Troch en Denys Slyvka bedanken voor de morele steun die mij heeft blijven motiveren, en de bruikbare feedback die ik heb verwerkt in deze thesis.

Samenvatting

Secrets management is een belangrijk onderdeel voor de beveiliging van gegevens. Het impliceert naar applicaties en methodes om gevoelige gegevens te beheren voor gebruik in applicaties, services en andere gevoelige delen van het IT-ecosysteem. Het moet maar één keer gebeuren, dat gegevens met administrator rechten tot een bepaalde server in de verkeerde handen vallen om daarna pijnlijke gevolgen te verdragen.

Bij Wolters Kluwer wordt TeamCity, een continuous integration & continuous development (CI/CD) tool, gebruikt voor automatisch applicaties op te bouwen met behulp van een chronologische set van taken die worden opgegeven. Dit onderzoek is vertrokken van een use case met betrekking tot deze tool. Het doel is om gevoelige gegevens beter te beheren in de CI/CD omgeving van TeamCity. Bij het uitvoeren van integratietesten en implementaties, gebruiken build scripts gegevens om toegang te verkrijgen tot externe servers en services. Traditioneel, worden wachtwoorden als veilige parameters op de TeamCity server opgeslagen. Dit biedt vaak niet een hoog genoeg beveiligingsniveau aan. In een productie omgeving van TeamCity met meer dan tienduizend build configuraties waar mogelijk honderden wachtwoorden opgeslagen staan, is het beheer van deze gegevens niet overzichtelijk. Secret management tracht dit probleem op te lossen aan de hand van twee tools die werden opgezet, een on-premise Hashicorp Vault opstelling, en een Microsoft Azure cloud oplossing, Azure Key Vault. Deze tools werden opgezet en geïntegreerd met TeamCity om gegevens te verlenen wanneer deze opgevraagd worden door build scripts.

Eerst werd een literatuurstudie uitgevoerd om de stand van zaken rond secrets management en cloud oplossingen te verduidelijken. Bij de methodologie werden open-source secret management tools bekeken aan de hand van de MoSCoW-methode. Bij De proof of concept kunt u de stappen volgen hoe beide opstellingen opgesteld zijn voor Wolters Kluwer, een wereldwijde leverancier van professionele informatie, software en diensten.

Vanuit dit onderzoek werd het duidelijk dat secret management tools gehanteerd kunnen worden om dit en gelijkaardig problemen op te lossen. Gevoelige gegevens die in een TeamCity omgeving staan, kunnen via secret management tools, centraal beheerd worden. Dit vormt een extra abstractielaag tussen de CI/CD tool en gevoelige gegevens. Secret management systemen bieden tegenwoordig veel functionaliteiten aan waarvan er zeker genoeg ruimte is om deze allemaal te onderzoeken in eventueel toekomstige onderzoeken.

Inhoudsopgave

Lijst van figuren

1. Inleiding

Bedrijven moeten meer de focus leggen op het beheer van gevoelige data zodat deze ten alle tijden veilig behouden worden. Wanneer applicaties en configuraties in een bedrijf draaien zullen er in bestanden uiteindelijk data komen die er niet zouden mogen zijn. Deze data zou beheerd moeten worden want men weet niet wie bijvoorbeeld binnen een week of twee ditzelfde bestand kan bekijken. Dit soort data noemt men ***secrets***. Veel bedrijven falen om dit in te zien en eindigen uiteindelijk met een overvloed aan applicaties en scripts waar gebruikersnamen en wachtwoorden zonder encryptie staan. Deze situatie noemt men een ***secrets sprawl***. Deze twee termen komen in onderdeel 2.1 aan bod en worden daar verder uitgelegd.

Doorheen de jaren zijn er een aantal applicaties verschenen die dit probleem proberen op te lossen door alle secrets in een centrale locatie op te slaan. Vanuit deze centrale node kan men authenticatie verlenen wanneer secrets nodig zijn.

Een belangrijke workflow binnen software bedrijven is *continuous integration & continuous development (CI/CD)*. Hierbij worden continu applicaties opgebouwd die aangestuurd worden door een chronologische set van taken. Deze taken bevatten soms dan gevoelige gegevens die over het hoofd worden gezien. Elke gebruiker die toegang heeft om deze taken te beheren kan deze gegevens zien.

1.1 Probleemstelling

Bij Wolters Kluwer wordt TeamCity, een CI/CD tool gebruikt voor automatisch applicaties op te bouwen met behulp van een chronologische set van taken die worden opgegeven. Deze taken bevatten soms secrets die niet in encryptie staan. Dit is geen best practice en

zou voorkomen moeten worden. TeamCity biedt de mogelijkheid aan om gevoelige data te maskeren in de vorm van *password parameters* maar alsnog is dit geen optimale manier van werken. Bij Wolters Kluwer zijn er momenteel 12966 build configuraties aanwezig waar zeker secrets in allerlei plaatsen zijn opgeslagen. Dit vormt de probleemstelling dat geen deftig overzicht is van waar secrets staan. Mogelijks staan er ook secrets zonder encryptie in parameter velden / build scripts. Het beheren van secrets is een beveiligingsmaatregel die vaak over het hoofd wordt gezien. Er bestaan applicaties die als doel hebben om deze gevoelige data af te schermen bij de werking van automatisering, CI/CD applicaties en container omgevingen. Deze applicaties bieden een oplossing aan voor gecentraliseerd beheer van secrets. Omdat Wolters Kluwer een hybride aanpak neemt met cloud en on-premise, zal er voor beide omgevingen een tool worden gebruikt. Deze dienen als fundament om het nut van deze technologie waar te nemen, en om deze verder te integreren binnen het bedrijf.

1.2 Onderzoeksvraag

Bij deze bachelorproef horen enkele onderzoeksvragen waar een antwoord op wordt gezocht, deze zijn als volgt:

- Welke open-source applicatie kan er gebruikt worden om **secrets** te beheren?
- Welke cloud oplossing kan er gebruikt worden om **secrets** te beheren?
- Welke opstelling on-premise of via cloud oplossing, geeft een betere werking voor de use case met TeamCity?

1.3 Onderzoeksdoelstelling

Dit onderzoek heeft tot doel het belang aan te tonen van secrets management binnen IT. Voor een Proof of Concept worden twee kandidaten gekozen om uitgewerkt te worden waar één als cloud oplossing, en één on-premise. Deze proof of concept zou ook een begin moeten zijn om het secrets probleem aan te pakken dat zich voordoet bij Wolters Kluwer met de CI/CD tool TeamCity. Het vormt ook als handleiding om de genomen stappen te reproduceren. Deze beide opstellingen trachten het probleem op te lossen met de gegeven reëel use case, en dienen aan als startpunt om na deze thesis verder geïntegreerd te worden binnen het bedrijf.

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 secrets managements & cloud computing, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht om met de MoSCoW-methode, een kandidaat te selecteren voor de on-premise opstelling.

In Hoofdstuk 4 wordt een proof of concept opgezet met de gekozen tools. Deze tools worden geconfigureerd & geïntegreerd met TeamCity waar de concrete probleemstelling ligt.

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 Secrets Management

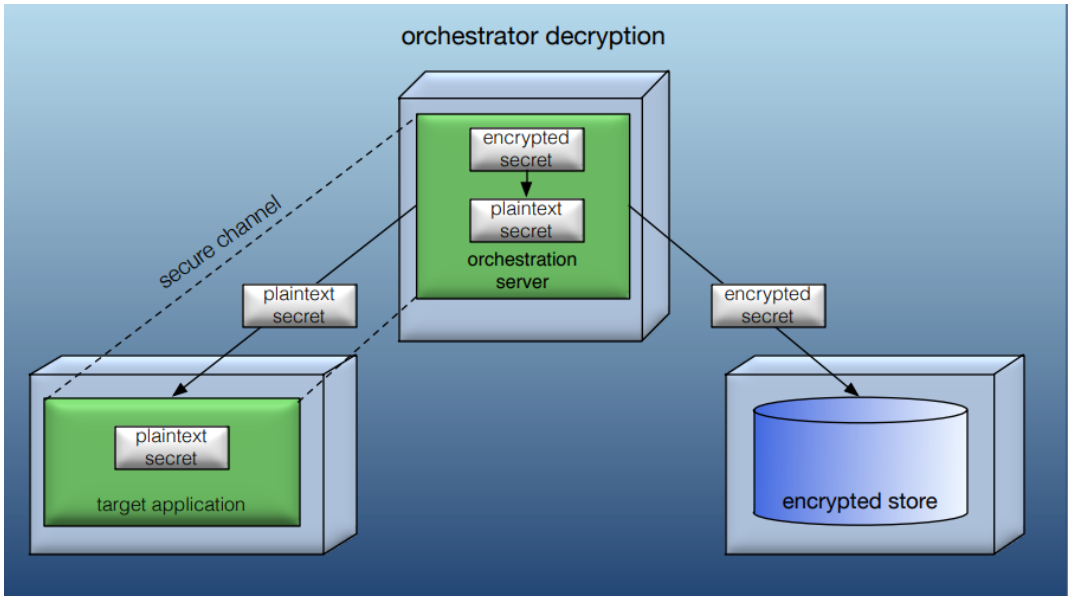
Om te verstaan wat **secrets management** inhoudt, worden de termen **secrets** en **secrets sprawl** weer verduidelijkt. Secrets impliceert een verzameling van gevoelige gegevens. Deze dienen als sleutel om digitale authenticatie uit te voeren, om toegang te krijgen tot een systeem (Dadgar, 2018). Enkele voorbeelden zijn:

- Gebruikersnaam en wachtwoord
- API tokens
- SSH keys
- Database inloggegevens

Deze gegevens zullen een secret sprawl veroorzaken wanneer deze niet onderhouden worden op de juiste manier. Secret sprawls ontstaan wanneer sensitieve gegevens, in allerlei plaatsen gebruikt worden om mee te werken. Bijvoorbeeld in automatisatie processen (Ansible, Puppet, Chef), containerized applications (Kubernetes, Red Hat OpenShift, Docker), Continuous Integration/Continuous Deployment (CI/CD) (TeamCity, Jenkins). Vroeg of laat belanden bestanden met onbewerkte tekst van gevoelige data in versiebeheer systemen, applicatie systemen of worden deze als lokale variabele gebruikt. Veel problemen ontstaan wanneer een bedrijf veel te laat beseft dat ze met een secret sprawl zitten, maar één van de grootste problemen is dat het bedrijf in meerdere locaties gevoelige gegevens heeft staan en dat er geen duidelijk overzicht is van waar deze gegevens zitten. Het bedrijf zal dan ook niet weten welke personen toegang hebben om deze gegevens te bekijken (Tozzi, 2020) (Dadgar, 2018).

Secrets management streeft naar een situatie waar secret sprawls vermeden worden, de gegevens geëncrypteerd zijn, dat de distributie van deze gegevens beheerd worden vanuit een gecentraliseerde locatie en als laatste dat deze gegevens gebruikt kunnen worden bij automatische processen zonder enige manuele tussenkomst. Somerfield (2015) legde uit hoe de uitdagingen die deze problemen stelden, werden aangegaan. Hier wordt het *Orchestrator Decryption* model uitgelegd (Somerfield, 2015).

2.1.1 Orchestrator Decryption



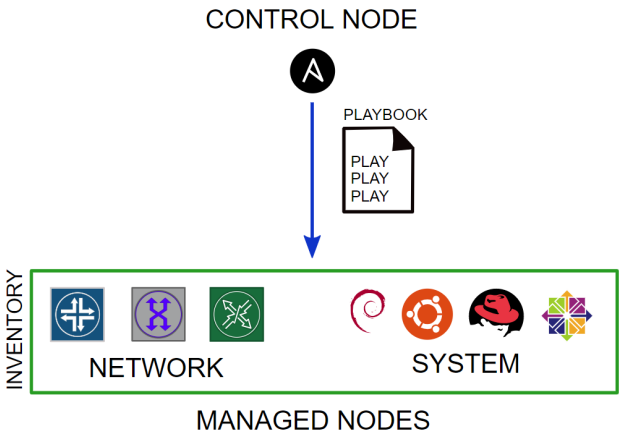
Figuur 2.1: Orchestrator Decryption (Somerfield, 2015)

Bij **orchestrator decryption** kijken we vooral naar *configuration management tools* zoals ansible, chef, puppet. Deze applicaties kunnen met een tussenkomst van bijkomende applicaties gevoelige data veilig houden wanneer deze niet gebruikt worden, deze zijn in rust geëncrypteerd. Wanneer de geëncrypteerde secret wordt opgeroepen bij een opdracht wordt deze door de orchestration tool naar een *plaint text* secret gedecrypteerd, met andere woorden wordt deze omgevormd tot onbewerkte tekst. Een voorbeeld van een orchestration georiënteerde tool is ansible. Deze tool kan gebruik maken van ansible vault om een vorm van secret management aan te gaan maar wordt gelimiteerd door een aantal factoren waardoor het binnen het model van *Orchestrator decryption* blijft (Somerfield, 2015). Om dit duidelijk uit te leggen, wordt in het algemeen uitgelegd wat ansible is en waarvoor deze tool gebruikt wordt. Verder wordt er ook uitgelegd hoe ansible vault hierop inspeelt.

2.1.2 Ansible

Ansible[1] is een open-source, configuration management tool (CMT) die door Red Hat[2] onderhouden wordt sinds 2015. Ansible is tegenwoordig op de voorgrond als het om geautomatiseerd configuratie, coördinatie en beheer van computer systemen gaat. Tegenwoordig is ansible de meest gebruikte CMT waar het applicaties zoals Chef, Salt en Puppet overtreft (Rayome, 2019). Het is ook handig voor simpele gebruikers die alledaagse taken willen uitvoeren binnen simpele netwerken, en niet alleen op één computer. Deze tool geeft je de mogelijkheid om binnen een complex netwerk van computers, vanuit één centraal punt, configuraties te deployen en systemen te beheren. Dit lost problemen op zoals bijvoorbeeld, wanneer men op verschillende computers het zelfde doel wilt bereiken zonder deze taak manueel opnieuw en opnieuw te doen op elke computer. Men kan via een centrale computer opdrachten versturen naar één of meerdere locaties tegelijkertijd. Hierdoor wordt ook het concept van menselijke fout vermeden.

In Ansible wordt er een onderscheid gemaakt tussen de control node en de managed node. De control node is een computer die ansible draait waar alle taken uit verzend worden. Deze taken voeren wijzigingen uit op de managed nodes. Ansible's manier van werken is allereerst een verbinding vast te leggen met de managed nodes, daarna wordt tijdelijk een programma gepushed dat uitgevoerd wordt tijdens deze verbinding. Deze programma's noemen ansible modules. Bij het einde van een taak wordt de module lokaal op de node verwijderd en sluit ansible de verbinding met de node. Ansible is *agentless* gebaseerd. Dit wilt zeggen dat op de managed nodes geen ansible applicatie aanwezig moet zijn die de configuraties verkrijgt en verwerkt. Ansible maakt gebruik van *connection plugins* waaruit **SSH** (Secure Shell) de meest gebruikte is. Deze worden gebruikt om een verbinding vast te leggen met linux / unix hosts. Voor Windows hosts wordt **WinRM** (Windows Remote Management) gebruikt (RedHat, 2021b).



Figuur 2.2: Werking ansible met control en managed nodes (CodingPackets, 2019)

[1] Ansible website
[2] Red Hat website

Omdat Ansible een tool omtrent automatisatie is, heeft deze tool dan ook instructies nodig. De instructies die ansible uitvoert zijn idempotent. Dit wilt zeggen wanneer bij een instructie een opdracht wordt uitgevoerd, maar er valt niets te wijzigingen in het systeem, zal ansible hier niets doen. Ansible zal deze stap overslaan en de volgende opdracht proberen uitvoeren (CodingPackets, 2019). Deze instructies worden als taken opgenomen in een **YAML** (Yet another Markup Language) bestand (Ingerson, g.d.). Een jaar na de introductie van het YAML formaat is de betekenis van het acroniem verandert naar (YAML Ain't Markup Language) (van Vugt, g.d.), dat is inderdaad een recursief acroniem.

Figuur 2.3: Voorbeeld inventory file ansible

```
[web]
be-dev-ansible-01
be-dev-ansible-02

[web:vars]
ansible_python_interpreter=/usr/bin/python3
ansible_ssh_user=John
ansible_ssh_pass=JohnSecure123
```

De inventaris afgebeeld op figuur 2.3, impliceert dat de machines, *be-dev-ansible-01* en *be-dev-ansible-02*, met de extra variabelen opgeroepen kunnen worden als groep web. Hierdoor gebruiken playbooks volgende python versie, volgende SSH gebruikersnaam en het bijbehorende wachtwoord bij het uitvoeren van volgende taken.

Figuur 2.4: Voorbeeld ansible playbook

```
1.  ---
2.  - hosts: web
3.    tasks:
4.      - name: permit traffic on port 10000-10001/tcp
5.          firewalld:
6.          port: 10000-10001/tcp
7.          permanent: yes
8.          state: enabled
9.      - name: ensure nginx is at the latest version
10.       apt: name=nginx state=latest
11.     - name: start nginx
12.       service:
13.           name: nginx
14.           state: started
```

In figuur 2.4 wordt er geïllustreerd hoe ansible voor de groep web, volgende taken gaat uitvoeren. Allereerst wordt er gezien of de TCP poorten 10000 en 10001 open staan, indien dit het geval is gebeurt er hier niets en gaat het playbook verder. Dit wordt aangeduid door het *state: enabled* gedeelte. Verder wordt er gekeken om nginx te installeren via de

command-line utility commando apt, als nginx reeds geïnstalleerd is zal er gekeken worden of die van de laatste versie is. Na het uitvoeren of het overslaan van deze taak wordt er voor gezorgd dat nginx gestart is als dit niet het geval is. Moest er bijvoorbeeld iets intern mis gaan bij een van de taken, zal deze in het output scherm als een *fatal error* verschijnen met een boodschap waarom dit gebeurt is en voor welke host.

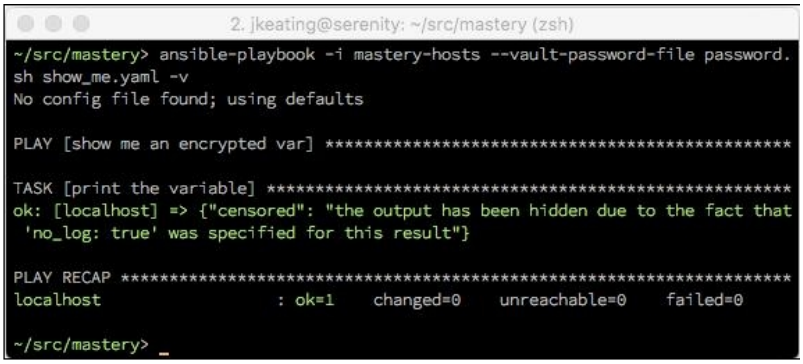
In het vorige voorbeeld werd er aangetoond hoe problematisch het is wanneer **secrets** in een productieomgeving in allerlei bestanden als onbewerkte tekst terechtkomen. Wanneer er meerdere playbooks worden aangemaakt die elk gevoelige data bevatten om de automatisatie te behouden. Is het onvermijdelijk dat deze vroeg of laat op source control terechtkomen. Eens dat gebeurt, is het al te laat. Via audit kan men vorige versies bekijken van *commits* en gevoelige data in de vorm van onbewerkte tekst zien (Wehner, 2015). Iedereen die in deze *public* of *private repository* zit kan de secrets zo bekijken, en men weet niet wie deze gegevens reeds bekeken heeft.

2.1.3 Ansible Vault

In het geval van ansible en playbooks biedt ansible vault een mogelijke oplossing, Ansible Vault[3] is een tool voor ansible dat bestanden met gevoelige data encrypteert. Dit biedt de mogelijkheid aan zodat alle taken kunnen uitgevoerd worden tijdens ansible taken die gegevens gebruiken die niet zichtbaar mogen zijn. Ansible zal de bestanden automatisch decrypteren tijdens looptijd, mits de sleutel voorzien is. Via één of meerdere wachtwoorden kan men inhoud encrypteren en decrypteren (RedHat, 2021a). Dit lost het probleem op dat gegevens niet meer zichtbaar zijn voor de buitenwereld. Maar ansible vault lost jammer genoeg het probleem niet volledig op. Wanneer ansible modules verstuurt worden naar een managed node, worden deze tijdelijk opgeslagen in een verborgen directory. Hier staan deze gegevens zonder encryptie. Dit wilt zeggen dat de gebruikte data, bijvoorbeeld wachtwoorden, overgezet worden op het bestandssysteem van de remote host. Zoals eerder verteld kan ansible alternatieve connection plugins gebruiken om een verbinding met een remote host vast te leggen. Als men SSH gebruikt zal de data geëncrypteerd en veilig zijn tijdens het versturen van de modules. Dit beveiligt de data tegen potentiële **sniffing attacks**. Dit is een tactiek die gebruikt wordt om data te achterhalen tijdens netwerk connectiviteit. Zo kunnen gegevens in onbewerkte tekst achterhaald worden (Passi, 2018). Eens data verzonden is, kan het zijn dat deze toch op de remote host geschreven worden zonder enige encryptie. Dit kan bijvoorbeeld gebeuren wanneer de omgevingsvariabele **ansible_keep_remote_files** gebruikt wordt. Hiermee blijven gebruikte modules op de remote host staan. Moest deze omgevingsvariabele niet gebruikt worden kan het nog altijd zijn wanneer een verlies van connectiviteit zich plaatsvindt, de modules op de remote host niet verwijderd zijn. Deze blijven daar tot ze manueel verwijderd worden (GB, 2018). Verder is er ook nog de mogelijkheid om de omgevingsvariabele **no_log** te gebruiken en deze op *true* te stellen. Dit zorgt er voor dat wanneer de modules weergeven worden in het systeem log, de gevoelige data verborgen blijft. Dit wordt afgebeeld op figuur 2.5 (GB, 2018).

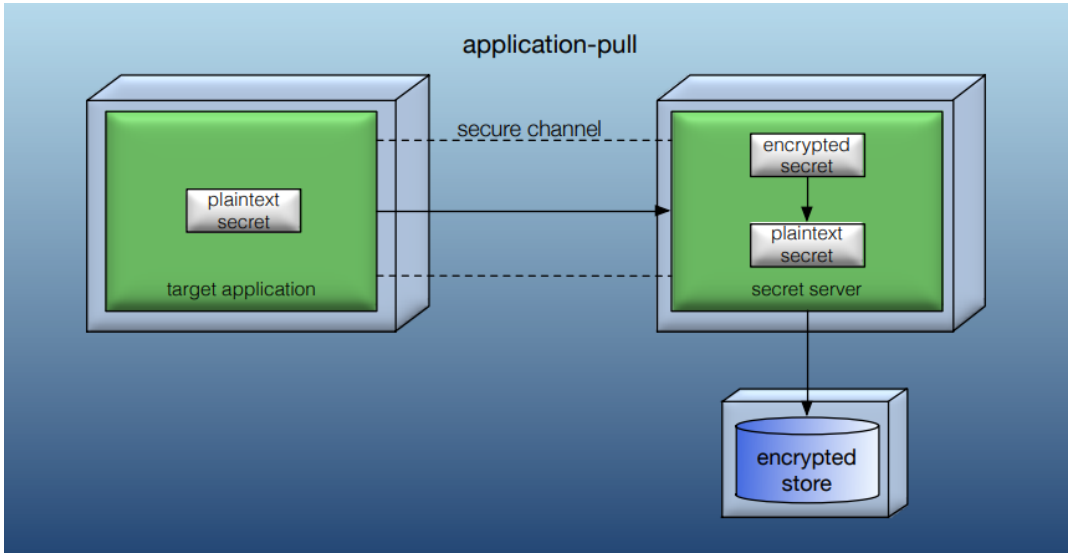
[3] Ansible Vault website

Zoals bij ansible vault getoond werd, is het een last om er voor te zorgen dat secrets niet gelekt worden bij het uitvoeren van taken en bij het beheren van bestanden met deze data. Het kanaal waarmee secrets verstuurd worden is niet altijd veilig genoeg om op te vertrouwen. Het beheer van alle secrets via de ansible vault manier is dan ook niet efficiënt in een ondernemingsomgeving. Er is geen vorm van rotatie zodat gegevens na zoveel tijd aangepast worden. Als een persoon een wachtwoord vergeet voor een bestand geëncrypteerd door ansible vault zijn die gegevens zo goed als verloren. De nadelen zijn hier veel te groot en het grootste probleem bij secrets management van secrets centraal te beheren is hier niet opgelost.



Figuur 2.5: gebruik van 'no_log: true' in ansible (GB, 2018)

2.1.4 Application-Pull



Figuur 2.6: Application-pull (Somerfield, 2015)

Een model dat meer aansluit bij het principe van secrets management is **application-pull**. Hierbij kan een persoon, of een service, zich via een systeem laten authenticeren om toegang te krijgen tot een locatie waar secrets beheerd worden. Via een beleid wordt er bepaald of toegang verleend wordt. Alles gebeurt nu ook vanuit een gecentraliseerd punt. Dit is een verbetering en een stevig concept dat meer aansluit bij secret management. Deze applicaties garanderen een aantal sterke punten, namelijk:

- *Central management*
- *Access policies*
- *Auditing*
- *Ephemeral credentials*
- *Compartmentalization*
- *Facilitates rotation*

Dit zijn belangrijke kenmerken voor een secrets management systeem. Vault van Hashicorp is één van de applicaties die *application-pull* duidelijk verwoordt. Volgens Somerfield (2015) was dit in 2015 een applicatie die al in de juiste richting ging (Somerfield, 2015). We zijn ondertussen, bij het schrijven van deze thesis, zes jaar verder.

2.1.5 Hashicorp Vault

Hashicorp, Inc.[4] is een softwarebedrijf opgericht door Mitchell Hashimoto en Armon Dadgar in 2012. Ze bieden open-source tools en commerciële producten aan die ontwikkelaars de mogelijkheid aanbiedt om applicaties op te zetten in een multi-cloud omgeving. Hashicorp staat bekend voor Vagrant en Terraform. Maar nu gaat de focus naar Vault. Vault is een **secret management tool** die ontworpen is om gecentraliseerd gevoelige gegevens te bewaren en toegang te beheren tot deze data in een omgeving met weinig vertrouwen. Het kan gebruikt worden om gevoelige data op te slaan en om dynamisch toegang te genereren voor andere applicaties en services (Lugger, 2020). Hashicorp is al jaren mee om een oplossing te zoeken tot de probleemstelling die gemaakt werd door **secrets** te beheren via een optimale manier. Via vault hopen zij het grootste deel van deze problemen aan te gaan via drie niveau 's (Dadgar, 2018).

Niveau 1: Central Management

In het eerste niveau wilt vault het **secrets sprawl** probleem tegengaan, hiermee worden alle **secrets** in een centraal punt verzameld. Deze gegevens worden gedecrypteerd en zo behouden gedurende de drie stadia van data, namelijk: *at rest*, *in transit*, en *in use*. Gegevens belanden niet meer in source control of dergelijke zaken, en staan veilig en gedecrypteerd in één locatie. Een volgend punt is de toegangscontrole die vergemakkelijkt wordt. In tegenstelling tot een bitbucket repository van een bedrijf waar iedereen toegang tot heeft, kan men beheren welke gegevens gezien en gebruikt kunnen worden door welke applicaties, alsook welke personen. Daarnaast is er ook een *audit trail* die gebruikt kan worden om na te gaan welke gegevens gebruikt zijn geweest door applicaties en personen (Dadgar, 2018).

Niveau 2: Dynamic Secret

Bij dit volgende niveau wordt de probleemstelling benaderd waarbij niet alle applicaties juist omgaan met secrets. Men kan optimaal het eerste niveau beheren maar toch secrets laten lekken door applicaties die deze gegevens gaan gebruiken. Applicaties kunnen bijvoorbeeld secrets loggen naar een log systeem, ze kunnen in diagnostische rapporten verschijnen of zelf opgenomen worden door monitor applicaties als deze actief zijn. Hieruit kan een besluit genomen worden dat applicaties een slechte werking kunnen hebben met secrets. Om dit probleem op te lossen maakt vault gebruik van *ephemeral credentials*. Dit zijn vluchtige gegevens die dynamisch aangemaakt kunnen worden. Met andere woorden zullen deze gegevens na een bepaalde duur niet meer geldig zijn. Een volgende stap is dat men deze gegevens gaat compartimentaliseren om duidelijker de bron van schade terug te vinden. Als voorbeeld heeft men 40 web servers die allemaal dezelfde gegevens gebruiken om een connectiviteit vast te leggen met een databank, een **shared secret**. Eens deze gegevens gelekt worden kan men moeilijk weten van waar deze gegevens gelekt zijn geweest. Als men voor elke web server unieke gegevens gebruikt, zal men weten dat er bijvoorbeeld een lek heeft plaatsgevonden bij webserver zeven. Hiermee kan men dan

[4] Hashicorp website

ook de gegevens die webserver zeven gebruikt intrekken en deze isoleren van de andere webservers. Indien alle webservers dezelfde gegevens zouden gebruikt hebben, zouden al deze webservers onderbroken worden. Met andere woorden *dynamic secrets* gaan boven *shared secrets* (Dadgar, 2018).

Niveau 3: Encrypt as a Service

Het laatste niveau dat wordt aangekaart is hoe externe applicaties die gebruik maken van vault, data verkrijgen, en deze data juist gebruiken wanneer deze *at rest* zijn. Men kan bijvoorbeeld kijken naar de situatie wanneer encryptie sleutels beheerd worden in vault. Deze sleutels worden aan applicaties verleend om cryptografie toe te passen op de data. Applicaties implementeren cryptografie vaak niet op een juiste manier, dit is ook iets dat Duong en Rizzo (2011) hebben aangekaart in een onderzoek waar ze aantonen hoe cryptografie misbruikt wordt in de security designs van een groot deel van het web. Hun focus ging naar ASP.NET. Vault heeft dit probleem waargenomen en stelde zich de vraag hoe ze het voorkomen om een verzameling van encryptie sleutels te beheren en deze te verlenen aan applicaties waar zij cryptografie uitvoeren. Dit is een concept dat is uitgegroeid tot een hele service genaamd *Encrypt as a Service*. Hier geeft vault de mogelijkheid om *named keys* aan te maken. Dit kunnen bank gegevens zijn, krediet kaart gegevens, een identificatienummer, ... Hier worden *High level API's* gebruikt om transactie 's uit te voeren met deze gegevens zoals encrypteer, verifieer, teken. Bij applicaties waar zulke transacties gebeuren, wordt vault via een API aangeroepen om deze zorgvuldig uit te voeren, zonder de nood van de *named keys* en hun gegevens te delen met de applicatie. De implementatie voor dit proces wordt behandeld door vault, men moet applicaties niet vertrouwen om *high level operations* zorgvuldig uit te voeren, en de key management gebeurt ook nog altijd door vault. Deze worden afgeschermd bij de andere applicaties (Dadgar, 2018).

Via vault werd er aangetoond hoe secrets management een vooruitgang heeft gehad voor secrets te beheren. Door de opkomst van cloud computing zijn er ook een tal secrets management systemen ontstaan in de vorm van cloud oplossingen. Azure Key Vault is een voorbeeld, van Microsoft Azure. Maar eerst wordt in het algemeen uitgelegd wat cloud computing is en de basis hier omtrent.

2.2 Cloud Computing

Volgens de ***national institute of standards and technology***[5] is cloud-computing (CC) een model, waar services aangeboden worden via een netwerk toegang (internetverbinding) naar een gedeelde netwerkpool van configureerbare ICT-middelen. Deze zijn makkelijk en snel lanceerbaar met het minimum aan moeite zonder enige tussenkomst met de service provider (Mell, Grance e.a., 2011). Dit impliceert dat applicaties, services en data door een eindgebruiker gebruikt kunnen worden via een gebruikersinterface (Malathi, 2011). Een heel simpel voorbeeld is het gebruik van e-mail services, zoals: Outlook, Gmail, ... Bij deze zaken heb je juist een computer met een internet verbinding nodig. Dan kan je gebruik maken van de e-mail services die worden aangeboden door een externe provider. Verder duidt het *national institute of standards and technology* aan dat het CC model opgebouwd is uit vijf essentiele karakteristieken, namelijk:

- On-demand self service
- Broad network access
- Measured service
- Resource pooling
- Rapid elasticity

Bij ***on-demand self service*** zullen de gebruikers met de gegeven resources automatisch kunnen werken zonder tussenkomst van de service provider. Verder zorgt ***broad network access*** ervoor dat de aangeboden services toegankelijk zijn via een netwerkverbinding. ***Measured Service*** volgt het principe dat de gebruiker betaalt voor wat men gebruikt, zo is er geen verspilling van resources. Dit wordt ook wel eens het ***pay as you go*** model genoemd. ***Resource Pooling*** volgt het principe dat service providers computing resources aan meerdere consumenten aanbieden met de noden van de klant voor het gebruik van virtuele en fysieke middelen. Het geeft een meerwaarde dat gebruikers de zelfde apparatuur gebruiken voor zowel gebruiker als provider. Als laatste is ***Rapid elasticity*** de eigenschap dat een service *elastisch* moet zijn wat impliceert dat een service moet kunnen op- en afschalen om aan de gegeven capaciteit te voldoen. Met andere woorden is het belangrijk dat er bij piekbelasting moet kunnen worden opgeschaald zonder enige moeite om voor geen storingen te zorgen, verder bij rustige perioden moet er dan ook weer kunnen worden afgeschaald. In kort zijn dit de vijf hoofd kenmerken van cloud computing. Het model is ook nog opgebouwd uit drie **service models** en vier **deployment models** die later in deze thesis aan bod komen (Mell, Grance e.a., 2011).

[5] Nist website

Tegenwoordig kent de wereld een aantal grote spelers bij het aanbieden van cloud oplossingen waaronder:

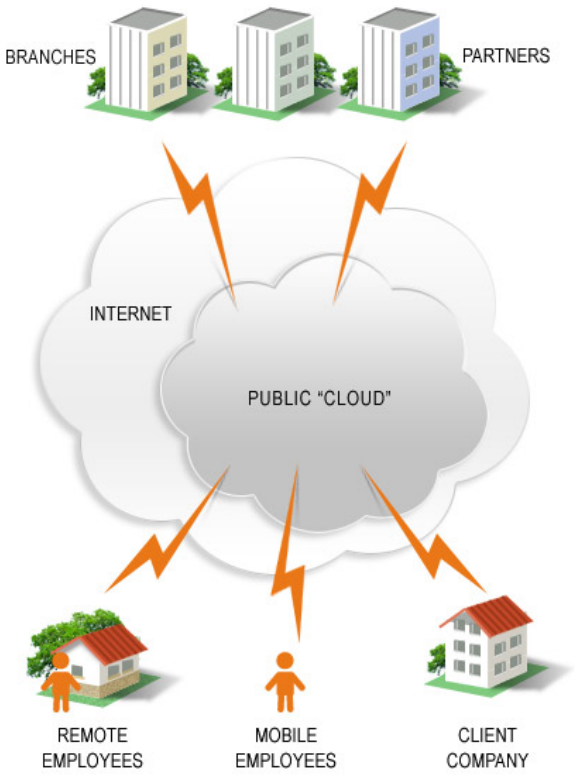
- Amazon Web Services (AWS) [6]
- Microsoft Azure [7]
- Google Cloud [8]
- Alibaba Cloud [9]
- Digital Ocean [10]

Dit zijn voorbeelden van cloud providers die een public cloud omgeving aanbieden. Hier betaalt de gebruiker voor het gebruik van de ter beschikking gestelde resources. Cloud providers zoals AWS en Microsoft Azure hanteren het eerder besproken concept van ***Measured Service*** zelfs. Ze bieden hun services aan volgens een ***Pay as you go*** subscriptie model zodat men betaalt voor wat men uiteindelijk gebruikt. Zo worden bijvoorbeeld bedrijven niet aangerekend voor resources die ze niet gebruiken op het moment zelf. maar ze hebben wel de toegang om deze te gebruiken. Dit is handig wanneer bijvoorbeeld doorheen de dag veel gebruik gemaakt wordt van bepaalde resources maar tijdens de nacht de curve daalt. Verder proberen cloud providers de ***Quality of Service*** optimaal te houden tegenover de eindgebruiker in een dynamische omgeving. Door de voortdurende groei en optimalisatie van cloud computing komen telkens nieuwe mogelijkheden aan bod voor ontwikkeling van applicaties en systemen (Mell, Grance e.a., 2011). CC zorgt er uiteindelijk voor dat bedrijven kunnen opereren zonder een ouderwetse infrastructuur. Natuurlijk zijn er mogelijkheden waarmee er gecombineerd kan worden met reeds bestaande infrastructuur. Hierdoor noemt het ook cloud **oplossingen**, voor elk probleem en elke situatie is er wel een oplossing.

[6] Amazon AWS website
[7] Microsoft Azure website
[8] Google Cloud website
[9] Alibaba Cloud website
[10] Digital Ocean website

2.2.1 Deployment Models

Public cloud



Figuur 2.7: *public cloud figuur (mizitechinfo, 2013)*

Eerder werd er gesproken over een aantal cloud providers. Deze bieden namelijk diensten aan, die als eerder vermeld een ***Pay as you go*** gebaseerde subscriptie volgen. Gebruikers betalen bijvoorbeeld voor de **CPU cycles**, of met andere woorden het gebruik van processor krachten, gebruikte opslag of de gebruikte bandbreedte. Het grootste voordeel hiervan is dat bedrijven zich geen zorgen moeten maken om uitbreidingskosten of het beheren van deze infrastructuur. Er moet niet worden gekeken om een locatie te voorzien, om deze machines te laten draaien want dit wordt al gedaan voor de eindgebruiker. Indien er toch wordt uitgebreid voor meer rekenkrachten of als dit juist verminderd wordt, is dit maar binnen enkele muisklikken te doen (Sullivan, 2015). Takken zoals beveiliging wordt door de cloud provider optimaal verzorgd. Dit is natuurlijk ook een risico dat gebruikers en bedrijven op zich nemen omdat hun gegevens door een externe partij beheerd wordt. Voor veel bedrijven is dit een serieuze afknapper samen met het feit dat de resources gedeeld worden met andere gebruikers (Castle, 2019). Wanneer een bepaalde cloud provider het slachtoffer wordt van een gegevens lek, is de kans natuurlijk groot dat de gegevens van de gebruikers gelekt worden. Dit is geen prettige scenario wanneer gevoelige data geëncrypteerd zou moeten zijn. Door gebruik te maken van een public cloud is dit iets wat bedrijven niet altijd onder controle hebben voor wanneer dit evenement plaats vindt. Door het gebruik van andere **deployment models** kan dit wel nog tegengewerkt worden.

Private Cloud

In tegenstelling tot public cloud waar bedrijven hun data niet volledig in eigen beheer hebben, is dit bij private cloud wel het geval. Hierbij worden cloud computing services aangeboden ofwel via het internet of via een intern privé netwerk uitsluitend aan bepaalde gebruikers. Dit staat niet open voor andere consumenten. Zoals bij de public cloud worden dezelfde services en mogelijkheden aangeboden. In deze instantie met de extra controle die beschikbaar gesteld worden, die via een computerinfrastructuur, die on-premise wordt gehost, gebruikt kan worden. Voor de beveiliging kan het niveau van diepgang zelf gekozen worden door de bedrijven dankzij zowel interne firewalls als interne hosting. Zo zijn de acties en gegevens afgeschermd tegenover externe providers. Private clouds worden vooral gebruikt door grote bedrijven om makkelijk toegang te bieden tot de gegevens over het hele netwerk van het bedrijf. Er zijn twee service modellen die geleverd kunnen worden binnen private cloud. Als eerste kan ***Infrastructure as a Service (IaaS)*** gebruikt worden waar infrastructurele resources ter beschikking gesteld worden. Verder kan ***Platform as a Service (PaaS)*** ook gehanteerd worden waar allerlei toepassingen door het bedrijf geleverd kunnen worden via het netwerk. In een verder stuk worden deze twee service modellen verder uitgelegd (Microsoft, 2021) (Castle, 2019) (Goyal, 2014).

Natuurlijk ontstaan er ook enkele voordelen en nadelen bij het hanteren van een private cloud. Zo zijn de kosten bij het opzetten van een private cloud hoger dan bij het gebruik van een public cloud door het kopen van de infrastructuur en de locatie voorzieningen hiervan. De software en de tewerkstelling van personeel voor de taken om deze te beheren en te onderhouden. Een van de belangrijkste voordelen is dan wel de veiligheid van de privacy van data, er zijn geen externe partijen betrokken bij het opereren van de data binnen het bedrijf.

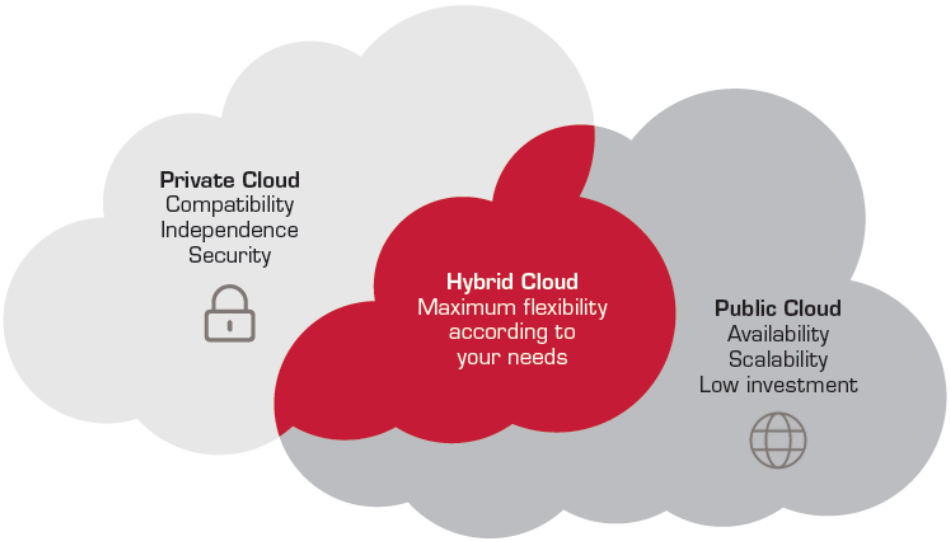
Community Cloud

Community cloud is een vorm waar niet te veel in wordt verdiept. Het volgt een **deployment model** waar het principe geldt dat een cloud computing oplossing wordt toegewezen aan een kleine *community* met dezelfde business doeleinden, door een andere *community*, of een externe partij, of een combinatie van beide gevallen (Jiménez e.a., 2013). Dit kan bijvoorbeeld een gezamenlijk project zijn waar *bedrijf a* een ander bedrijf, *bedrijf b*, een bepaalde service voor zich laat uitwerken zodat *bedrijf a* op andere zaken kan focussen. Zoals een webshop waar het gedeelte voor af te rekenen onderhouden wordt door een ander bedrijf.

Hybrid Cloud

Hybrid cloud is een vorm waar meerdere cloud vormen worden gebruikt om samen te werken. Zo kan een gedeelte private cloud gebruikt worden voor cruciale data waar andere minder gevoelige data in een public cloud kan gezet worden. Verder levert hybrid cloud voor een aantal problemen zoals het initieel opzetten. Het is een vereiste om een uitgebreide kennis te hebben over IT en analyse van bedrijfsprocessen. Verder ligt de complexiteit voor efficiënt gebruik hoger en kan het hierdoor voor een hogere werkdruk zorgen op

vlakken zoals de middelen wanneer geen deftige management tool gebruikt wordt om deze te implementeren of beheren (Sacic, 2020).



Figuur 2.8: *Hybrid cloud figuur (Sacic, 2020)*

2.2.2 Service Models

Zoals men eerder schreef wordt cloud computing gevormd door de vijf hiervoor genoemde en beschreven karakteristieken. Verder ook door de vier **deployment models** waar net over verteld werd en nog drie andere **service models** waar nu naar gekeken zal worden. deze bestaan namelijk uit:

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)

SaaS

SaaS is één van de vormen van **service models** waar de cloud provider applicaties ontwikkelt, onderhoud en deze ter beschikking stelt aan de gebruiker in de vorm van een ***pay as you go*** model zoals eerder besproken. De cloud provider neemt het hele onderhoudt van de infrastructuur en beveiliging op zich. Via een computer en een internet verbinding kan er gebruik worden gemaakt van de software (oracle, 2021) (Mell, Grance e.a., 2011).

PaaS

Bij dit model krijgt de klant de mogelijkheid applicaties te laten lopen op de systemen en ook de keuze om op deze cloud infrastructuur te werken zonder enige zorg aan onderliggende zaken zoals: netwerk, servers, besturing en opslag. Los van deze zaken heeft de gebruiker wel nog altijd autorisatie over de configuraties en beheer van de gelanceerde applicaties en gegevens (Mell, Grance e.a., 2011).

IaaS

De mogelijkheid wordt hier voorgesteld bij de gebruiker voor verwerking van data, opslag, netwerken en andere fundamentele computer middelen zodat de gebruiker zelf de keuze heeft om zijn noden te voldoen. De gebruiker beheert de onderliggende cloud infrastructuur niet maar heeft wel controle over de besturing systemen, opslag en gelanceerde applicaties. Verder is er ook de mogelijkheid om de netwerk componenten te configureren voor gelimiteerde toegang (Mell, Grance e.a., 2011).

2.2.3 Secrets Management als cloud oplossing

Azure Key Vault

Azure Key Vault[11] is een PaaS oplossing dat je toelaat om beveiligd cryptografische sleutels te beheren. Secrets worden onderhouden via deze service. Wanneer bijvoorbeeld, virtuele machines en schijven worden opgezet in de cloud, worden deze door Azure zelf geëncrypteerd. De mogelijkheid is er om zelf een encryptie sleutel te gebruiken om de schijven van virtuele machines te encrypteren. Deze encryptie sleutels kunnen dan via de Azure Key Vault bijgehouden worden. Deze tool laat gecentraliseerd beheer toe voor secrets.

Via het platform kan een secret worden aangemaakt met de kenmerken die eerder werden opgesomd bij het *application-pull* model:

- Central management
- Access policies
- Auditing
- Ephemeral credentials
- Compartmentalization
- Facilitates rotation

De secret wordt centraal beheerd via een vault dat kan worden aangemaakt via het Azure platform. Via *Identity Access Management (IAM)* kan er beheerd worden waar een gebruiker toegang tot heeft en welke acties deze gebruiker heeft uitgevoerd (Ganatra, 2019). Dit wordt verder ondersteund door het activiteitenlogboek dat Azure aanbiedt binnen de key vault service. Als secrets worden aangemaakt wordt er de keuze gegeven of er een activatie of expiratie moment wordt opgegeven. Dit zorgt er voor dat gegevens vluchtig kunnen zijn. Men kan meerdere vaults maken met meerdere gegevens die voor allerlei applicaties gebruikt worden. Deze worden geleverd aan applicaties via service principals. Dit zijn applicaties dat binnen Azure Active Directory worden aangemaakt om toegang te verkrijgen tot resources in Azure. Deze kunnen gebruikt worden om secrets op te halen voor applicaties. Secrets zijn nooit hard coded en kunnen binnen de key vault opnieuw gegenereerd worden. Deze worden bijgewerkt en applicaties verkrijgen zo geautomatiseerd hun bijgewerkte sleutel. Er zijn meerdere manieren om sleutel generatie en rotatie te automatiseren na een bepaald tijdsinterval. De meest gebruikte manier is via Azure Automation met een geautomatiseerd script (Majumder, 2019) (Marczak, 2020).

[11] Microsoft Azure Key Vault website

3. Methodologie

In dit hoofdstuk wordt er gekeken naar de mogelijke applicaties en wordt een keuze gemaakt aan de hand van de MoSCoW-methode, deze methode gebruiken we om de kenmerken te lijsten in categorieën om sneller in te zien welke kandidaat meer geschikt is. Dit verdelen we onder vier categorieën waaronder:

- Must have
- Should have
- Could have
- Won't have

3.1 Criteria voor keuze tools

3.1.1 Must have

In dit gedeelte kijken we naar de absolute must bij een applicatie, zo moet de applicatie aan volgende vereisten voldoen:

- Application-pull kenmerken
- Open-source
- GUI aanwezig (met TLS verbinding)
- Ondersteuning voor Windows (OS)
- Ondersteuning TeamCity

Er wordt een focus gelegd op een gratis applicatie waar men via een GUI, beveiligd met een TLS verbinding, centraal de secrets kan beheren. Er moet ook een vorm van beleid

zijn zodat niet iedereen aan elke gegevens kan. Een integratie met TeamCity is in dit geval belangrijk zodat deze gebruikt kan worden bij de proof of concept. Verder is het ook belangrijk binnen Wolters Kluwer dat het centrale platform beveiligd moeten kunnen worden met een getekend certificaat. Dit is een standaard voor platformen binnen het Wolters Kluwer netwerk voor veiligheidsredenen.

3.1.2 Should have

Het zou ook geen te ingewikkeld proces mogen zijn om de opstelling te maken. Dit moet evident op te zetten zijn zodat later andere mensen hiermee zonder veel moeite mee kunnen werken.

3.1.3 Could have

Een integratie met LDAP zodat het secret management systeem met active directory gegevens werkt is een pluspunt. Hiermee wordt er de mogelijkheid gegeven om via *active directory user accounts*, binnen het domein van Wolters Kluweren, authenticatie te verrichten. In een bedrijf zoals Wolters Kluwer wordt LDAP gebruikt wanneer er integraties mogelijk zijn met applicaties.

3.1.4 Won't have

Een volle integratie met verscheidene applicaties zal niet aan bod komen maar voor de concrete use case met TeamCity wel. Men gaat geen complexe configuraties uitvoeren om alle functionaliteiten te gebruiken.

3.2 Keuze tools

3.2.1 On-premise

- Vault
- Knox
- Confidant
- 1Password Secrets Automation

Vault

Vault[1] is een secrets management applicatie die in deel 2.1.5 eerder besproken werd. Los van alle technische vereisten waar alle specificaties van komen wordt vault ook voor Windows ondersteund en gebruikt een duidelijke web interface. Vault maakt gebruik van

[1] Vault website

een persistente backend om geëncrypteerde data in te behouden. Hiervoor kan Consul [2] gebruikt worden. Bij een opstelling zou er voor een simpele backend gekozen worden zoals een lokaal filesystem storage backend. Vault is open source en kan als een managed vault via cloud geleverd worden. TeamCity heeft één plugin [3] voor een integratie met Vault dat gebruikt kan worden. Verder is er ook de mogelijkheid voor LDAP te integreren met Vault, zo kunnen AD gebruikers zich aanmelden via de web interface en volgens policies gegevens beheren.

Knox

Knox [4] is een applicatie aangemaakt door Pinterest omdat ze problemen hadden met het juist beheren van secrets. Zo bewaarden ze vroeger secrets in source control. Dit zorgde uiteraard voor een secret sprawl. De groei van het aantal ontwikkelaars zorgde ook voor bijkomende risico's zoals malware en phishing indien gegevens lekten. Knox zorgt voor een centraal beheer van secrets. Het geeft de mogelijkheid gebruikers toegang te verlenen wanneer secrets nodig zijn en het systeem ondersteunt de mogelijkheid voor rotatie van de gegevens. Er wordt ook bijgehouden wie welke gegevens gebruikt heeft aan de hand van audit. Knox kan in zowel Linux als MacOS en Windows omgeving worden opgezet samen met de installatie van *GO*[5], de taal waar Knox mee geschreven is. Knox zal ook gebruik moeten maken van een backend. Standaard maakt deze gebruik van een *TempDB* die in geheugen alle data zal opslaan. Maar dit is geen werkwijze om permanent te behouden. Zou de machine uitvallen is alle data verloren. Voor de Backend wordt MySQL, PostgreSQL of sqlite aangeraden (Lundberg, 2016). Knox heeft verder geen community achter zich die integraties met andere applicaties aanbieden. De algemene setup van deze werking is ook zeer complex en is deze geen goede keuze voor een bedrijf zoals Wolters Kluwer.

Confidant

Confidant [6] is een secret management systeem ontworpen door Lyft. Het wordt gebruikt voor exclusief gebruik binnen een AWS omgeving. Als backend voor deze applicatie wordt een DynamoDB gebruikt wat een AWS databank systeem is. Het opzetten van deze opstelling is heel technisch en kan alleen worden gedaan binnen een Docker omgeving of een Linux omgeving. Via Docker kan men deze tool op Windows gebruiken maar er wordt geen gebruik gemaakt van containers. Voor Confidant is er weinig tot geen community die achter het platform staat (Confidant, g.d.).

[2] Consul website
[3] JetBrains website met Hashicorp Vault plugin
[4] Knox github pagina
[5] GO website
[6] Confidant github pagina

1Password Secrets Automation

1Password Secrets Automation [7] is een recent uitgekomen secret management system gemaakt door AgileBits Inc die beter bekend staan als de ontwikkelaars van 1Password. In 2021 hebben ze een grote stap gezet door SecretHub[8] aan te kopen waarmee ze hun stap richten naar secrets management (Schoonen, 2021). De tool gebruikt de beveiliging en encryptie mechanismen van 1Password wat een succesvolle tool is die sinds 2006 bestaat. De tool ondersteunt het principe van centraal beheer voor alle secrets met een belang aan audit, key rotation, policy en integraties met reeds bestaande tools (Shiner, 2021). Voor TeamCity is er (nog) geen plugin. Deze tool is ook geen open-source applicatie. Omdat de applicatie bij het schrijven van deze tool vrijwel net uitgekomen is, is er ook niet veel informatie over op het internet bij het gebruik en mogelijke plus, en minpunten.

Keuze applicatie

In onderdeel 2.1.5 werd Vault van Hashicorp gebruikt om de uitleg te geven hoe *application-pull* een model is dat zeer sterk secret management definieert. Vault is na veel jaren verder uitgegroeid tot een zeer geschikte applicatie om als secret management system te gebruiken. Deze zal uitgewerkt worden op een Windows systeem.

3.2.2 Cloud oplossing

Wolters Kluwer gebruikt voor het grootste deel Azure als hun cloud solution platform. Hierdoor zal er gekeken worden naar het opzetten en gebruik van Azure Key Vault. Dit is een cloud oplossing die eerder werd besproken en bij Wolters Kluwer al een tijdje interessant leek te zijn om te gebruiken. Juist was er nooit de tijd om hiervoor een project voor op te zetten. Deze zal worden gekozen als cloud oplossing. Voor TeamCity bestaan er enkele plugins die een integratie voeren met Microsoft Azure cloud oplossingen, waaronder één plugin[9] voor Azure Key vault.

[7] 1Password Secrets Automation website
[8] SecretHub website
[9] JetBrains website met Azure Key Vault plugin

4. Proof of Concept

In dit hoofdstuk worden de opstellingen met omgevingen voor Hashicorp Vault en Azure Key Vault opgezet. Alle vermelde scripts / playbooks zijn aanwezig op volgende github repository[1].

4.1 Opzet Hashicorp Vault

De applicatie wordt opgezet in een virtuele test omgeving, on-premise van het Wolters Kluwer development infrastructuur. Een aantal zaken die aanwezig waren alvorens het begin van de proof of concept (PoC).

- Dell laptop binnen het Wolters Kluwer domein met WSL² Ubuntu + ansible 2.10.6
- Virtuele test omgeving binnen Wolters Kluwer domein
- API test call script met dank aan de co-promotor
- Administrator rechten bij CI/CD tool TeamCity in productie en development

Windows Subsystem for Linux is een compatibiliteitslaag voor het native draaien van binaire linuxbestanden in een console omgeving op een Windows 10 machine (whitewaterfoundry, g.d.). Deze wordt gebruikt om Ansible te laten draaien. Verder zijn de specificaties van de virtuele test omgeving als volgt:

- OS: Windows Server 2019 Datacenter (64 bit)
- Intel Xeon Silver 4208 CPU @ 2.10GHz
- 32GB geheugen

[1] github repository

4.1.1 Installatie & configuratie Hashicorp Vault

In onderdeel 2.1.2, werd getoond wat Ansible is en hoe deze tool gebruikt wordt om taken te automatiseren. Doorgaans deze PoC zal ansible gebruikt worden voor een aantal taken snel en efficiënt te voltooien. De syntaxis die gebruikt wordt om een Ansible script op te roepen in de WSL² Ubuntu omgeving is als volgt: *ansible-playbook PLAYBOOK.YML -i INVENTORY*. Bij het klaarzetten van de omgeving wordt Ansible gebruikt aan de hand van playbook *installVault.yml*[2] met bijbehorende inventory file[3].

Installatie

Alvorens het starten, vraagt het playbook om de gebruikersnaam op te geven met bijbehorend wachtwoord. Dit is zo bij elk playbook gedaan zodat we deze gegevens niet rechtstreeks in de inventory file zetten zoals er afgebeeld werd op figuur 2.3. Eens het playbook afgerond is, kan vault opgestart worden via een elevated command prompt. Dit is een command prompt met administrator rechten. De working directory moet dezelfde zijn als de locatie waar *vault.exe* staat. Eens in deze locatie wordt de commando *vault server -config=.\config\config.hcl* gebruikt om vault op te starten met bijbehorende configuratie bestand.

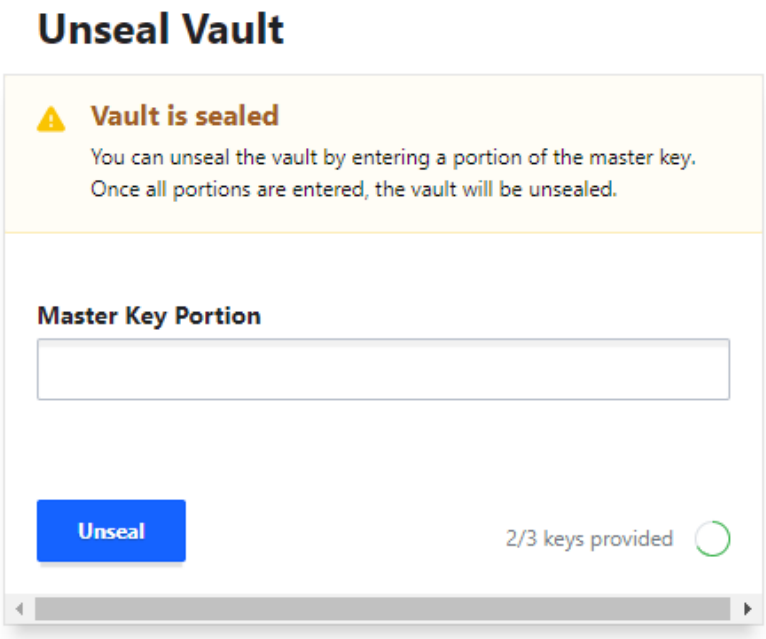
Figuur 4.1: inhoud initieel configuratie bestand

```
1.    ui = true
2.    disable_mlock = true
3.    storage "file" {
4.    path = "./data"
5.    }
6.    listener "tcp" {
7.    address     = "0.0.0.0:8200"
8.    tls_disable = "true"
9.    }
10.   api_addr = "http://127.0.0.1:8200"
11.   #cluster_addr = "https://127.0.0.1:8201"
```

Vault kan worden opgestart in *dev* modus waar de lokale CLI geauthenticeerd is om met vault te communiceren. basis configuraties zijn in deze modus verwerkt en dient als testomgeving. Data wordt (geëncrypteerd) in vluchtige geheugen opgeslagen. Bij het afsluiten van de vault applicatie is alle toegevoegde data verloren. Omdat we bij deze PoC data willen opslaan en de vault omgeving geregeld afgesloten moet worden, is hier gekozen voor een simpel bestandssysteem. In het configuratiebestand wordt dit vermeld. Vault zal de backend bestanden lokaal creëren en encrypteren. Elke keer wanneer men de vault wilt opstarten zal deze in een sealed state zijn. Wanneer vault sealed is verwacht hij de master key om encryptie uit te voeren. Wanneer men de webpagina **http://127.0.0.1:8200**

[2] installVault.yml op github
[3] inventory file op github

voor de eerste keer bezoekt, wordt een pagina weergeven om de master key op te delen in een aantal key shares. Verder wordt er ook opgegeven hoeveel key shares er nodig zijn om vault te openen. Deze key shares vormen het vermogen om de master key terug op te bouwen. Bij deze PoC kiezen we voor vijf key shares met drie nodige keys om de master key terug op te bouwen en de vault te openen. Vervolgens kan men een JSON bestand downloaden waar de vijf key shares in staan met een root token. Dit bestand moet zeer goed worden bewaard. Deze key shares zijn nodig om vault te openen wanneer deze sealed is. Indien deze key shares verloren gaan kan men vault op geen andere manier meer openen en is de data verloren! De bijbehorende root token wordt gebruikt om als root gebruiker te authenticeren. Dit is in het begin nodig om policies en gebruikers aan te maken. Verder moet het gebruik van de root key in een productieomgeving tot een minimum gehouden worden.



Figuur 4.2: proces om sealed vault te openen (Vault, 2021)

Configuratie

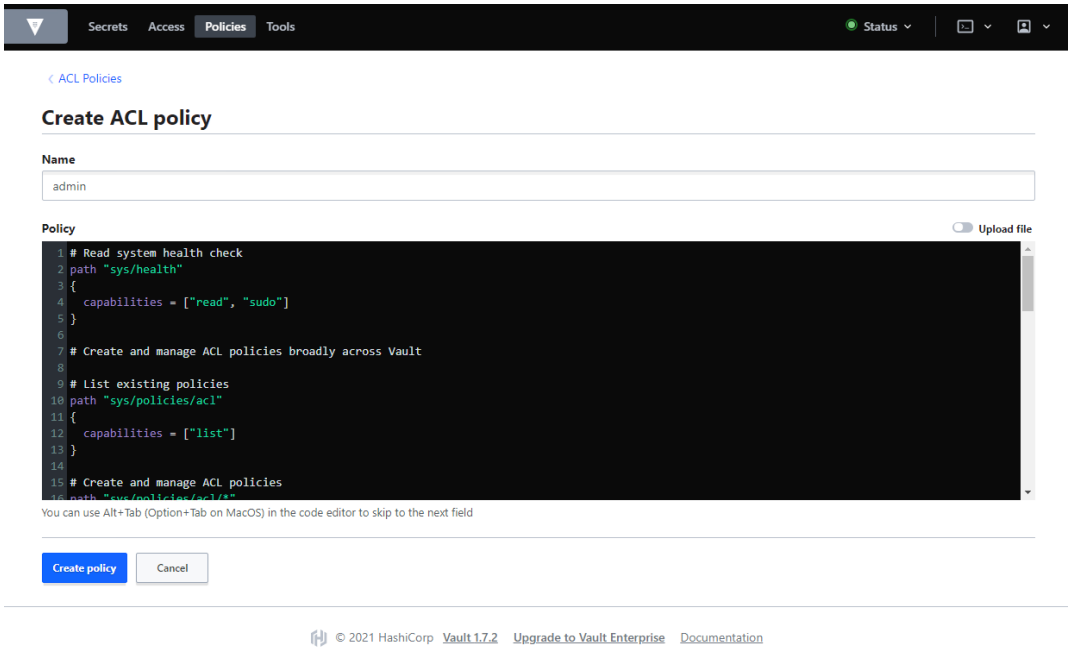
De root key wordt gebruikt om men aan te melden als root gebruiker. Nu men aangemeld is op vault kan men operaties verrichten binnen dit systeem. Hier worden in de volgende delen configuraties verricht:

- Secrets (Secrets Engines)
- Policies (ACL Policies)
- Access (Authentication Methods)

Allereerst wordt een nieuwe *secrets engine* aangemaakt. Een secrets engine is een component die data opslaat, encrypteert of decrypteert. Hier wordt er gekozen voor een kv secrets engine aan te maken. Alle standaard waarden worden behouden. Eens deze engine

aangemaakt is, kan men hier secrets opslaan met een key-value methode. Hier maken we een secret aan waar we later key-value gegevens aan toe voegen. Deze geven we de naam *TeamCity*. Hierin worden twee secrets aangemaakt, namelijk de keys *TeamCityAccount* en *TeamCityAccountEncryptedPassword* met bijbehorende waarden.

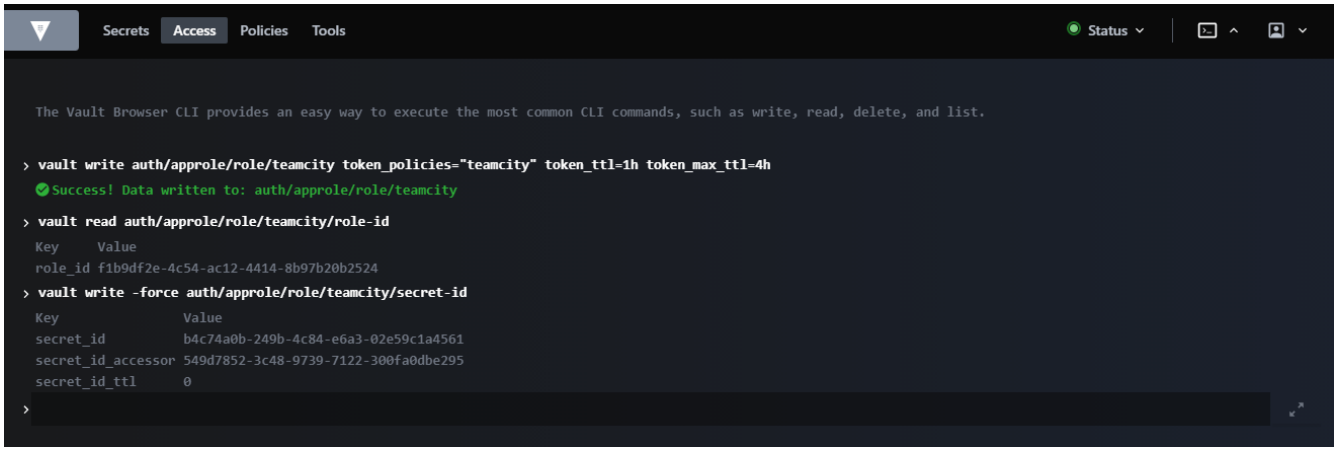
Vervolgens worden de policies aangemaakt. Hier worden 2 policies aangemaakt, namelijk *teamcity* en *admin*. Dit wordt verricht door een ACL policy aan te maken. Om dit te doen kunnen we via de web ui een policy definiëren. Deze kan ook worden opgeladen. Via ansible laten we het playbook createPolicyFiles.yml[4] lopen om twee HCL bestanden aan te maken waar de policies voor de beide rollen in gedefinieerd staan. Deze worden in de plugin directory gestoken. Voor admin wordt de naam *admin* opgegeven en bijbehorende HCL bestand wordt hierbij opgeladen. Ditzelfde gebeurt ook voor de teamcity policy met naam *teamcity*.



Figuur 4.3: proces voor policy aan te maken (Vault, 2021)

Nu wordt er gekeken naar de access tab. Hier kan men methodes toevoegen om authenticatie te verrichten. Men kan een *userPass* authenticatie methode toevoegen om met gedefinieerde gebruikersnamen met bijbehorende wachtwoorden, aan te melden op het systeem. Dit zal hier niet worden gedaan omdat in deel 4.1.2 een integratie wordt uitgevoerd met LDAP om AD gegevens te gebruiken binnen het domein van Wolters Kluwer. Er wordt nu wel al een *Approle* toegevoegd. Deze dient als authenticatie methode voor applicaties onder een bepaalde rol. Deze rol wordt aangemaakt zodat via de TeamCity integratie, later secrets kunnen worden opgehaald vanuit vault. De configuratie hiervoor wordt via de CLI van de vault UI uitgevoerd. De configuratie wordt weergeven met output waarden in figuur 4.4. Van deze waarden worden later de role_id en secret_id values gebruikt voor de plugin integratie met TeamCity.

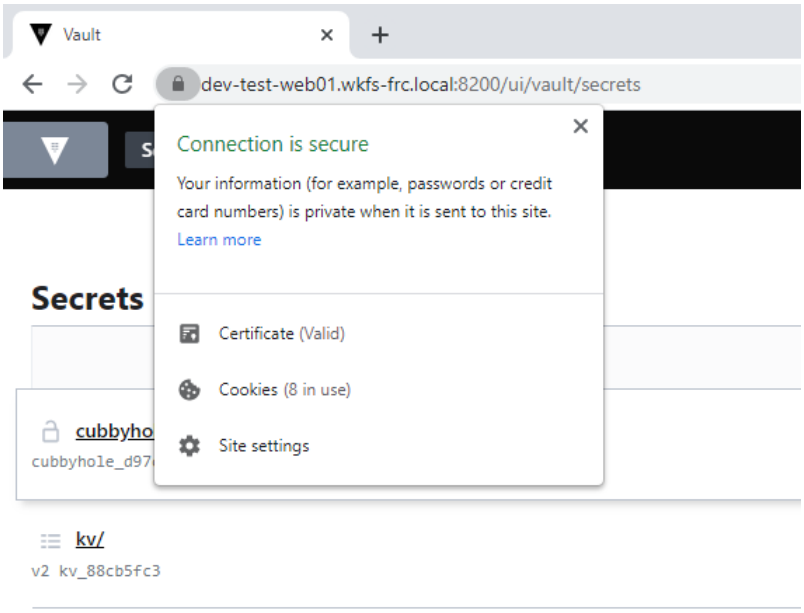
[4] createPolicyFiles.yml op github



Figuur 4.4: proces voor approle aan te maken (Vault, 2021)

4.1.2 LDAP integratie

In een vorig onderdeel werd er gesproken over de access tab waar authenticatie methodes worden toegevoegd. Nu wordt vault geïntegreerd met LDAP zodat gebruikers van het development team zich kunnen verbinden met vault als een gebruiker. Er wordt gekozen voor een nieuwe authenticatie methode. Hier wordt LDAP gekozen. De standaard waarden worden behouden en de methode wordt geactiveerd. Vervolgens komt de configuratie om vault met LDAP te verbinden. Hier wordt de URL gegeven van de domeincontroller: *ldap://EXAMPLE.local:389*. Vervolgens wordt bij User Attribute de waarde *samaccountname* gegeven. Zo kunnen gebruikers via hun gebruikersnaam aanmelden die ze gewoon zijn, bijvoorbeeld *John.Doe*. Vervolgens wordt het bindDn gegeven waarmee gebruikers opgezocht worden *CN=EXAMPLE-IT-LDAP,OU=Service Accounts,OU=EXAMPLE,DC=DC-EXAMPLE* en de userDn *DC=DC-EXAMPLE* met bijbehorende bindpass. Eens dit gelukt is kunnen gebruikers van Wolters Kluwer een verbinding maken met vault. Nu moet er een groep worden aangemaakt waar gebruikers aan toegevoegd worden voor admin rechten. Binnen de LDAP authenticatie methode, maakt men een groep aan genaamd *poc users* met de *admin* policy. Vervolgens kunnen gebruikers worden toegevoegd zodat deze de policy verkrijgen.

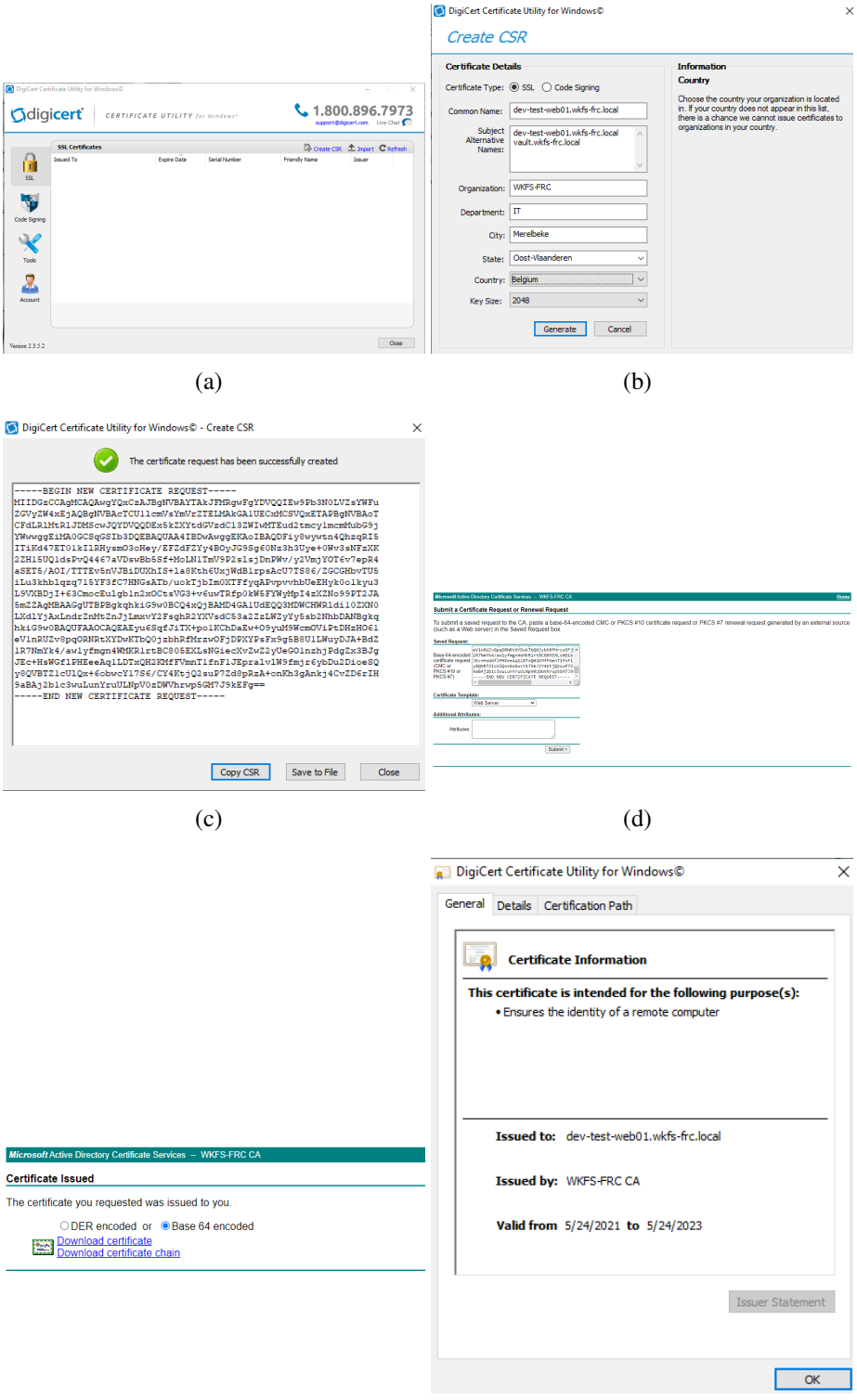


Figuur 4.5: Beveiligde webapplicatie (Vault, 2021)

4.1.3 Installatie SSL certificaat

In dit onderdeel wordt een SSL certificaat aangemaakt en toegevoegd aan de PoC. Dit is nodig voor het verkeer te beveiligen tussen clients en de vault webpagina. Binnen Wolters Kluwer is dit een requirement om aan te voldoen bij webapplicaties. Om dit te behalen wordt de applicatie *Digicert*[5] gebruikt. Hiermee wordt een certificaat aanvraag opgesteld die verder behandelt wordt door *Microsoft Active Directory Certificate Services*. Hier wordt de aanvraag verwerkt en verkrijgt men een *certnew.p7b* bestand. Hier is het certificaat in opgeborgen. Daarna gaat men terug naar de Digicert applicatie en wordt dit bestand geïmporteerd. Hieruit kan men vervolgens drie bestanden downloaden. Een *CAcert.crt* bestand en twee andere bestanden behorende tot de host waarvoor de certificaten getekend zijn. In dit geval is dat *dev-test-web01_wkfs_local.crt* en *dev-test-web01_wkfs-frc_local.key*. Deze drie bestanden worden in de map *certs* gestoken onder *config*. In het configuratie bestand moet er worden verwezen naar deze certificaten. TLS moet ook worden geactiveerd. Het adres **http://127.0.0.1:8200** zou ook niet meer gebruikt worden, het FQDN **https://dev-test-web01.wkfs-frc.local:8200** zal hier gebruikt worden. Vervolgens moet de omgevingsvariabele in het systeem ook verandert worden zodat deze ook het juiste adres zal gebruiken. Dit wordt allemaal behaald via het playbook updateConfigFile.yml[6]. Eens het configuratie bestand is bijgewerkt wordt vault opnieuw opgestart om deze te gebruiken. In figuur 4.6 kan men het proces bekijken hoe het resultaat van 4.5 behaald werd.

[5]Digicert website
[6]updateConfigFile.yml op github



(a) (b)

(c) (d)

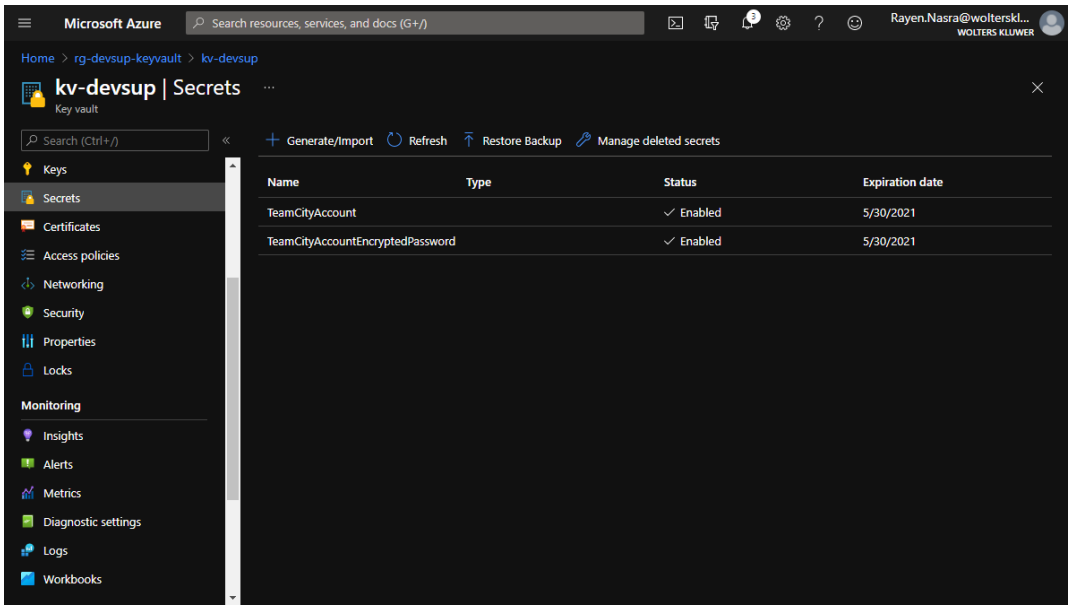
(e) (f)

Figuur 4.6: proces om SSL certificaat te tekenen (Digicert, 2021)

4.2 Opzet Azure Key Vault

Het opzetten van een Key Vault van Azure is minder complex dan een on-premise secrets management opstelling zoals bij Hashicorp Vault. Toch zijn er enkele stappen die uitgevoerd moeten worden voor secrets beheerd kunnen worden. Als eerste wordt een nieuwe resource group aangemaakt. Deze krijgt de naam *rg-devsup-keyvault*. Hier maken we een nieuwe resource. We kiezen voor key vault en noemen deze *kv-devsup*. De recent aangemaakte resource group wordt hier gekozen en de resterende opties worden niet gewijzigd. Hiermee bekomen we tot het resultaat afgebeeld in figuur 4.7.

Vervolgens wordt een *service principal* aangemaakt voor de key vault die gebruikt zal worden door de integratie plugin van Azure Key Vault en TeamCity. Dit wordt behaald via de Azure CLI met commando *az ad sp create-for-rbac –name TeamCityVault*. Vervolgens moeten we enkele waarden verkrijgen over deze service principal die later nodig zijn bij de integratie. Via volgende commando's wordt dit behaald: *az account show –query 'tenantId:tenantId,subscriptionid:id';* en *az ad sp list –display-name TeamCityVault –query 'clientId:[0].appId'*. Deze commando's werden door een persoon verricht met bevoegdheden aan Azure AD binnen het Wolters Kluwer domein. De aangemaakte service principal wordt juist nog toegevoegd bij de gebruikers van de key vault. Dit gebeurt via de access policies tab binnen de *kv-devsup* key vault. Dit staat ook afgebeeld op figuur 4.7. Hier wordt de service principal *TeamCityVault* gekozen en toegevoegd met de **GET** key permission. De service principal moet juist de rechten hebben om secrets op te halen. Nu worden twee secrets aangemaakt in deze vault. Net zoals bij Hashicorp Vault, worden de keys *TeamCityAccount* en *TeamCityAccountEncryptedPassword* met bijbehorende waarden toegevoegd.



Figuur 4.7: Aangemaakte Key Vault in Azure (Azure, 2021)

4.3 TeamCity omgeving

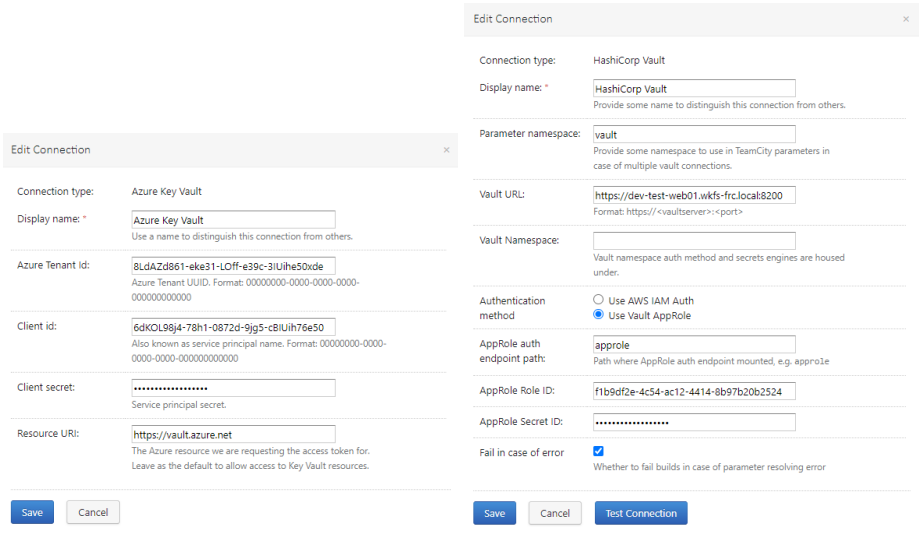
4.3.1 Integratie tools met TeamCity

In dit gedeelte worden de integratie plugins geïnstalleerd op TeamCity. Hierna worden connecties vastgelegd met de Azure Key Vault en Hashicorp Vault. Als eerste wordt het playbook *fetchPlugins.yml*[7] gebruikt om de beide plugins te downloaden en klaar te zetten in de vault directory onder *plugins*. Hierna worden deze op de webapplicatie van TeamCity opgeladen via volgende stappen:

1. Op het homescreen van TeamCity kiest men voor het tandwiel icoon (Administration)
2. Nu kiest men voor *plugins* onder *Server Administration*
3. Hier laad men elke plugin afzonderlijk op
4. De toegevoegde plugins worden ingeladen en zijn klaar voor gebruik

Er is een project aangemaakt dat dient als test case. Om hier de Azure Key Vault plugin en de Hashicorp Plugin te gebruiken moeten eerst connecties worden vastgelegd. Hier worden de eerder aangemaakte approle van vault, en de service principal van azure voor gebruikt. Dit wordt behaald via volgende stappen:

1. Het Secret Management project word gekozen onder *Development Support\Playground\Secret Management*
2. Hier kiest men voor *edit project settings*
3. Verder wordt onder *General Settings*, *Connections* gekozen
4. Hier worden voor zowel Hashicorp Vault als Azure Key Vault een connectie toegevoegd. Voorbeeld bij figuur 4.8



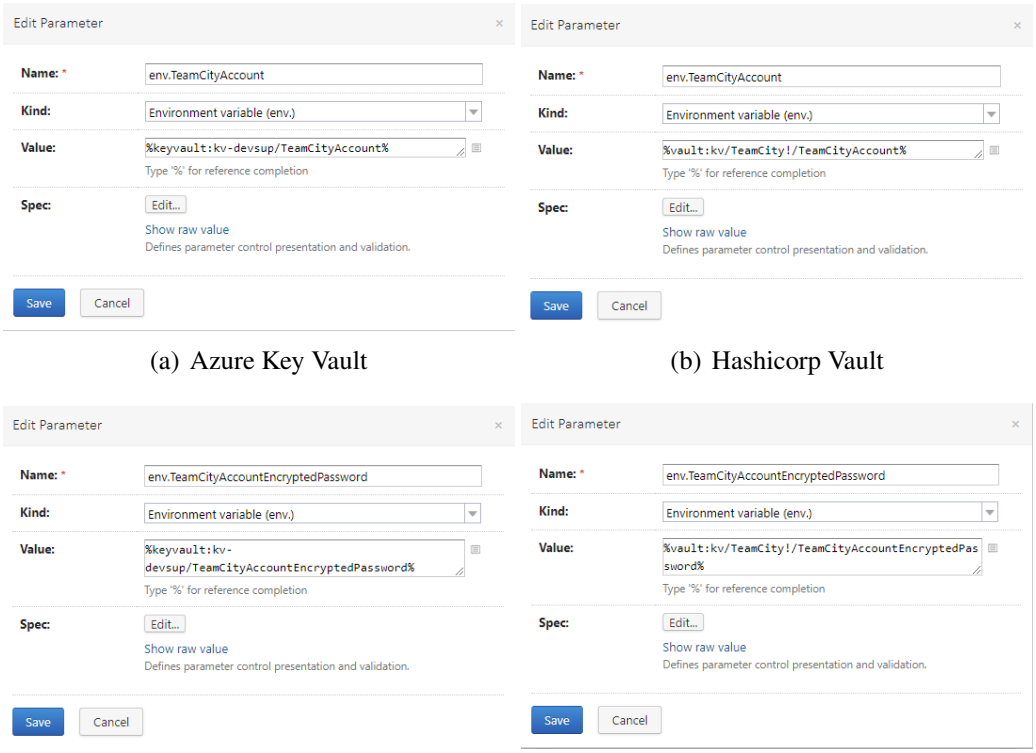
(a) Azure Key Vault (b) Hashicorp Vault

Figuur 4.8: proces connecties toe te voegen (TeamCity, 2021)

[7] fetchPlugins.yml op github

4.3.2 Secret Management Project

In TeamCity is er een project voorzien om meerdere build configuraties aan te maken. Voor Azure Key Vault en Hashicorp Vault worden twee verschillende build configuraties opgesteld onder hetzelfde project. De plugins doen hun werking aan de hand van omgevingsvariabelen die alvorens deployments van builds worden gedeclareerd. De plugins doen hun werking wanneer deze variabelen worden meegegeven als script argumenten. Hierdoor worden deze opgehaald en toegevoegd aan het proces van de build. Dit wordt behaald onder de configuratie opties van het project, onder *parameters*. Omgevingsvariabelen worden gedeclareerd met het prefix '*.env*'. Zo kan een variabele met naam *env.AgentsAmount* worden aangemaakt met een vaste waarde. Voor de Hashicorp Vault plugin geldt volgende syntaxis '**%vault:PATH!KEY%**'. Voor Azure key vault is dat '**%keyvault:VAULT NAME/SECRET NAME%**'. De waarden waar naar gerefereerd moet worden zijn de waarden die in deel 4.1.1 reeds toegevoegd zijn geweest bij hashicorp vault. Voor azure key vault werd dit gedaan in deel 4.2. Op figuur 4.9 staan voorbeelden waar deze variabelen worden toegevoegd.



(a) Azure Key Vault

(b) Hashicorp Vault

(c) Azure Key Vault

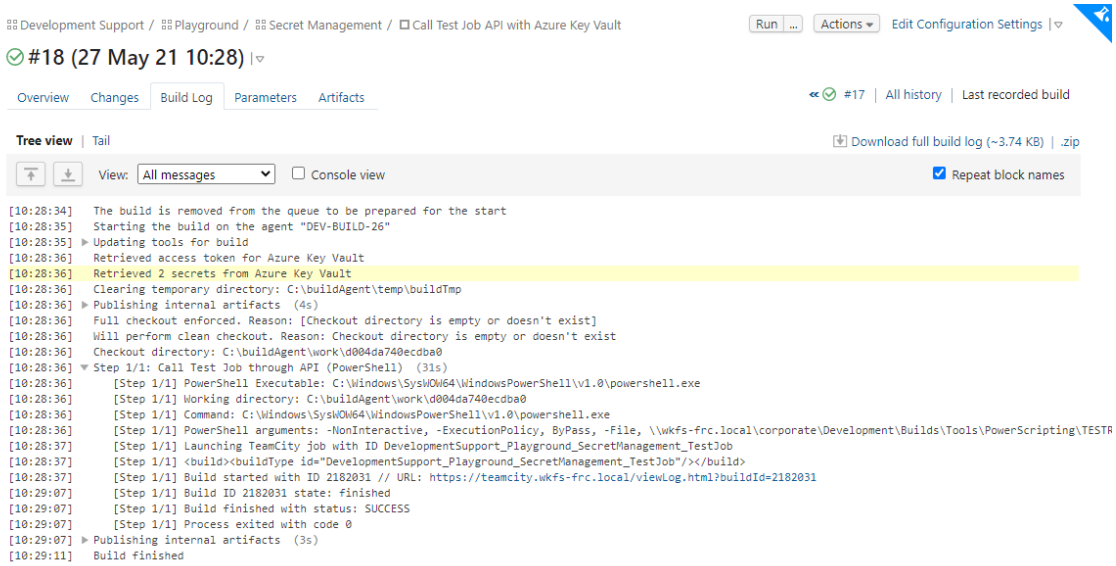
(d) Hashicorp Vault

Figuur 4.9: omgevingsvariabelen toevoegen (TeamCity, 2021)

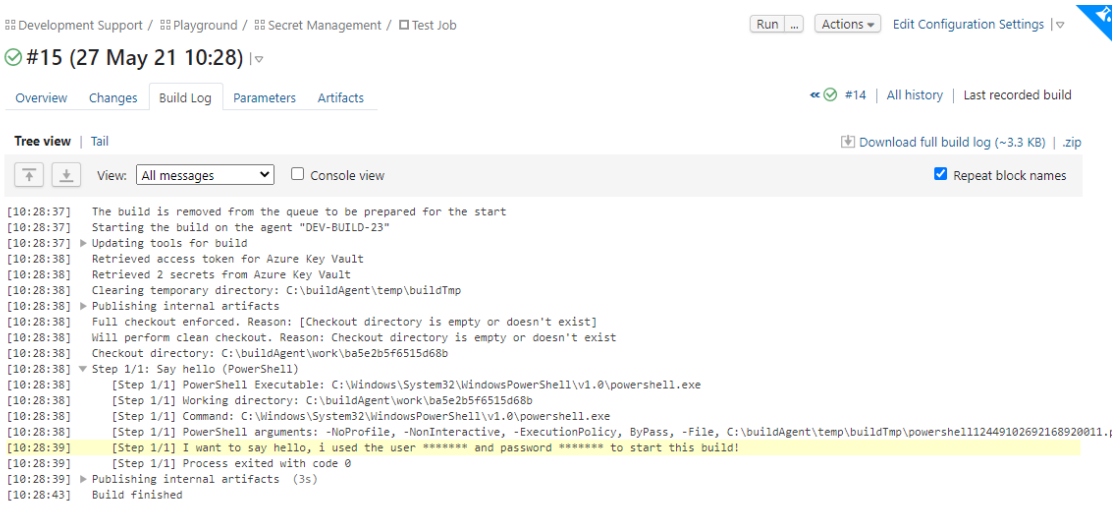
Voor beide build configuraties worden dezelfde build steps aangemaakt. Hier wordt een powershell script RunJobWithVault.ps1[8] opgeroepen waar de omgevingsvariabelen meegegeven worden. Deze worden toegevoegd aan het *Scrip argument* gedeelte als "**"%env.TeamCityAccount%"**" en "**"%env.TeamCityAccountEncryptedPassword%"**". Dit is het gedeelte waar de plugins de syntaxis van de variabelen zullen opmerken eens deze opgeroepen worden bij builds. Deze gegevens kunnen worden opgeslagen en verborgen gehouden in het TeamCity systeem waar ze gemaskeerd worden. Deze worden dus niet geëncrypteerd. In deze instantie worden de waarden geleverd door de vaults van Hashicorp en Azure. Dit voegt een extra abstractielaag tussen TeamCity en gebruikte secrets. Wanneer een build wordt opgeroepen wordt een bepaalde agent die inactief is, verwezen om de uitvoering te doen. Bij deze build wordt het script opgeroepen vanuit een netwerkshare en worden de omgevingsvariabelen meegegeven. Het script zal ervoor zorgen dat een API call naar de TeamCity server wordt gemaakt met gegevens van een administrator, waarmee een andere build wordt gestart. Deze tweede build heeft als enige werking een output te geven met de gebruikte secrets van de vorige build. Met andere woorden worden in beide scripts de secrets opgeroepen. In het tweede script wilt hij deze dan effectief in de output weergeven.

De werking van de Azure Key Vault build wordt in figuur 4.10 afgebeeld. Hier kan men opmerken dat de secrets via de eerder ingestelde connectie worden opgehaald. In de build agent wordt de access token in het vluchtige geheugen opgenomen waarmee toegang wordt aangevraagd aan Azure AD, gelimiteerd tot de key vault *kv-devsup*. De secrets worden geleverd als een wachtwoord parameter. De TeamCity server heeft nooit toegang tot de secrets en deze operaties worden meten gedelegeerd aan de build agent. (vyadh, 2018)

[8] RunJobWithVault.ps1 op github



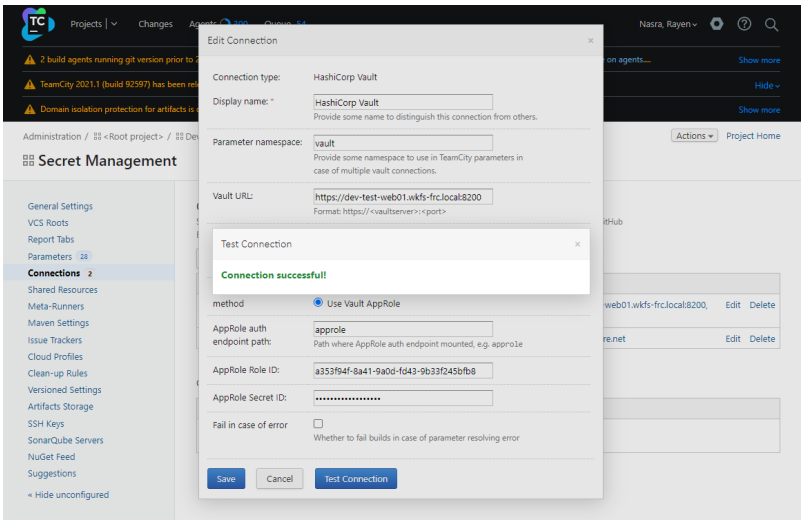
(a) API Call met secrets request



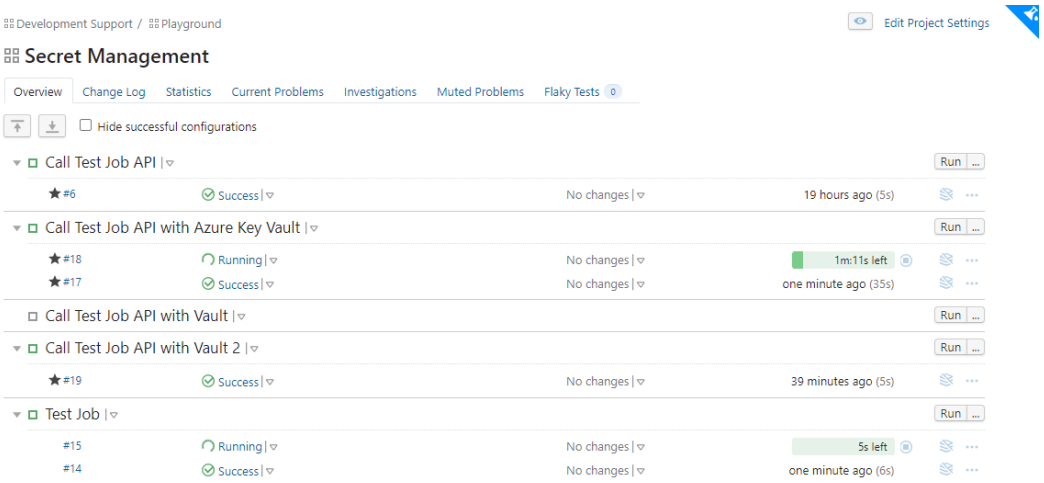
(b) secrets request en printen hiervan

Figuur 4.10: Build logs van azure key vault werking (TeamCity, 2021)

Voor Hashicorp Vault is deze werking niet gelukt. De configuraties van Vault zijn juist uitgevoerd en de approle is juist ingesteld voor TeamCity waarmee de connectiviteitstest succesvol voltooid, dit wordt afgebeeld in figuur 4.11. De plugin die gebruikt is faalt om de omgevingsvariabelen op te merken met de nodige gebruikte syntaxis. Waar het probleem bij ligt is niet meteen duidelijk maar het is wel duidelijk dat bij de builds geen logboekvermeldingen komen in verband met de Hashicorp Vault Plugin. Deze testen zijn gebeurt op zowel Windows als Linux agents en twee instanties van TeamCity. Als laatste wordt er in figuur 4.12 het secrets management project afgebeeld met lopende builds.



Figuur 4.11: Connectiviteitstest TLS verbinding in WK domein, met approle TeamCity (TeamCity, 2021)



Figuur 4.12: Secrets Management project in TeamCity met lopende builds (TeamCity, 2021)

5. Conclusie

Na het onderzoek kunnen we concluderen dat secrets management systemen zeker gebruikt kunnen worden bij automatiseren wanneer gevoelige data gebruikt wordt. Een complexe hiërarchie van secrets behouden voor elke automatisering tool, afzonderlijk van elkaar, is een zeer lastige taak. Bij aanvang van deze proef werden enkele onderzoeksvragen opgesteld, waar doorheen deze proef een antwoord op werd gezocht. Deze onderzoeksvragen zijn als volgt: " Welke open-source applicatie kan er gebruikt worden om **secrets** te beheren? ", " Welke cloud oplossing kan er gebruikt worden om **secrets** te beheren? " en " Welke opstelling on-premise of via cloud oplossing, geeft een betere werking voor de use case met TeamCity? ".

Op de onderzoeksvraag " Welke open-source applicatie kan er gebruikt worden om **secrets** te beheren? ", was Hashicorp Vault de meest interessante tool. De MoSCoW-methode werd gehanteerd om gelijkaardige tools te vergelijken op basis van een aantal kenmerken. Hashicorp Vault is ook de tool waarmee mijn interesse gewekt werd naar dit onderwerp. Het is een breed systeem die over de jaren heen de uitdagingen aanging die secret management aankaart. Voor de tweede onderzoeksvraag " Welke cloud oplossing kan er gebruikt worden om **secrets** te beheren? " werd de tool Azure Key Vault gekozen omdat Wolters Kluwer grotendeels Azure Cloud oplossingen hanteert. Bij Azure Key Vault lag ook meer de voorkeur voor gemak aan integratie door middel van zaken zoals Azure AD dat reeds gebruikt werd.

Van dit onderzoek werd verwacht dat de opstellingen niet te complex zouden zijn om op te zetten en voor Azure Key Vault was dit inderdaad het geval. Door het feit dat dit een cloud oplossing is, worden de stappen om deze te gebruiken, zo gebruiksvriendelijk mogelijk gehouden. Hier schuilen nog altijd technische competenties achter. Bij Hashicorp Vault was het opzetten iets meer complex door de reden dat deze volledig on-premise

was. Een LDAP integratie samen met een SSL certificering zet deze opstelling goed om direct gebruikt te worden door het development team. Enige verdere integraties met andere applicaties kunnen dan ook verwezenlijkt worden.

Op de laatste onderzoeksvraag " Welke opstelling on premise of via cloud geeft een betere werking voor de use case met TeamCity? " werd door middel van de proof of concept, een resultaat behaald, ook al is dit niet wat er in eerste instantie verwacht werd. De proof of concept is niet volledig gelukt. De integraties van TeamCity met de plugins van Azure Key Vault en Hashicorp Vault zijn gelukt. Voor Azure Key Vault zijn de test builds die werden opgesteld geslaagd met de juiste & verwachte werking. Secrets worden aangeroepen vanuit een beveiligde locatie waar deze centraal beheerd worden. Deze gegevens worden tijdelijk gebruikt zonder dat ze op de TeamCity server, noch de agent opgeslagen worden. Voor Hashicorp Vault is deze zelfde werking niet gelukt. De integratie was succesvol voltooid maar secrets worden niet opgeroepen. Na enige troubleshooting zou het probleem boven water kunnen komen.

Secrets management systemen bevatten veel domeinen die in de proof of concept niet aan bod zijn gekomen. Door de functionaliteiten die ter beschikking gesteld zijn, kan het zeer interessant zijn om deze verder te configureren en integraties te voeren met andere applicaties. Verder kan het ook zeker interessant zijn om deze functionaliteiten in de toekomst verder te onderzoeken.

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 Introductie

Het is een herkenbare situatie bij bedrijven wanneer gevoelige data zoals gebruikersnamen en wachtwoorden, extensief gebruikt worden bij automatisatie gebieden. Het probleem hiermee is dat deze gegevens makkelijk verspreid geraken in allerlei bestanden en uiteindelijk terecht komen op *version control* systemen zoals github [1] of bitbucket [2]. Deze manier van werken zorgt na een tijd voor een overvloed aan gevoelige data in meerdere locaties, zonder enig idee van wat waar is. Dit noemt men een **Secret Sprawl** (Tozzi, 2020). Dit is qua beveiliging geen prettige situatie en men weet niet welke personen deze data kunnen bekijken, en of deze personen bevoegd zijn. **Secret Management** is de term die gebruikt wordt die dit probleem de baas probeert te zijn. Het gaat om een gecentraliseerde locatie waar **secrets** beheerd en verleend worden aan applicaties en gebruikers die deze nodig hebben om taken uit te voeren (Hoffman, 2021). Dit brengt een niveau van veiligheid en abstractie omhoog terwijl de integratie met applicaties vlot gebeurd.

Dit onderzoek en de opstellingen die worden aangemaakt zijn binnen het kader van het development team binnen *Wolters Kluwer* [3] waar de bruikbaarheid van secrets management bewezen wordt en een integratie wordt uitgevoerd met Teamcity, een CI/CD applicatie.

[1] https://github.com
[2] https://bitbucket.org
[3] https://www.wolterskluwer.com/nl-be

Verder wordt dit proces geautomatiseerd tot waar mogelijk. Hieruit volgen volgende onderzoeksvragen:

- Welke open-source applicatie kan er gebruikt worden om **secrets** te beheren?
- Welke cloud oplossing kan er gebruikt worden om **secrets** te beheren?
- Welke opstelling on-premise of via cloud oplossing, geeft een betere werking voor de use case met TeamCity?

A.2 State-of-the-art

Secret management verwijst naar hulpmiddelen en methodes om digitale authenticatie en autorisatie tot systemen te beheren. Dit houdt in dat data zoals sleutels, wachtwoorden, API tokens, bevoegde accounts en dergelijke gevoelige data niet zomaar eindigen in onbewerkte teksten. deze data noemt men **secrets**. Het gebruik van secrets in meerdere locaties voor bepaalde doeleinden te behalen gaat tegen de beveiligingsnormen die standaard gelden binnen IT maar alsnog geschonden worden. Men weet op deze manier dan ook niet welke gebruikers toegang hebben tot bestanden waar lichtgevoelige data aanwezig is. Hiermee wordt er ook niet bijgehouden in welke locaties al die gevoelige data kan zitten. Via secrets management probeert men dit probleem aan te pakken. Enkele applicaties die het concept van secrets management ondersteunen zijn:

- Vault
- Keywhiz
- Confidant

Volgens de **Verizon Data Breach report (2020)** waren 77% van de cloud inbreuken gerelateerd met gecompromitteerde inloggegevens, met andere woorden, *secrets* (Hoffman, 2021). Dit impliceert naar een zwakke focus voor secret management terwijl bedrijven hun beveiliging systemen optimaal proberen te houden. Verder is er ook de vraag of er cloud oplossingen zijn om dit probleem aan te pakken? Maar om dit eerst te verstaan wordt het concept van cloud oplossingen uitgelegd.

Cloud oplossingen zijn services aangeboden door **Cloud Service providers** om bepaalde problemen op te lossen via het internet. Gebruikers krijgen computerdiensten die veel positieve kenmerken met zich meeneemt, onder andere:

- kostenefficiënt
- schaalbaarheid
- veiligheid van data
- flexibiliteit

Qua kosten-efficiëntie kan er gekeken worden dat er geen geld meer besteed moet worden aan een IT-infrastructuur en hiervoor moet geen locatie voor worden voorzien. De kosten dalen door het feit dat er ook niets meer onderhouden moet worden en niet moet gekeken worden naar opschalen waarbij geen extra apparatuur aangeschaft moet worden. Via een *Cloud Service Provider* kan er makkelijk opgeschaald of afgeschaald worden

naargelang de situatie en welke noden er voldaan moeten worden. In vergelijking met zelf een IT-infrastructuur te beheren is hier het grote voordeel dat men juist betaalt voor de functionaliteit van het apparatuur. Deze providers houden hun cybersecurity optimaal en investeren hier veel geld in, zo hebben de klanten altijd hun *Cloud Resources* ter beschikking zonder dat daar omtrent veel zorgen over zijn. Het enige dat vereist wordt bij de klant, is dat er een internetverbinding aanwezig is. Wat tegenwoordig een standaard hoort te zijn. Dergelijke zaken zoals het juist laten werken van de machines en beveiligingsmaatregelen wordt volledig opgenomen door de cloud service provider (Hoeffnagel, 2020). Om het concept van cloud oplossingen verder te begrijpen wordt er ook uitleg gegeven over de verschillende *Cloud Delivery Models*, namelijk:

- 'Infrastructure as a Service' (IaaS)
- 'Platform as a Service' (PaaS)
- 'Software as a Service' (SaaS)

Bij **IaaS** wordt het bij een eindgebruiker mogelijk gesteld om rekenkrachten en opslag te huren zonder dat zij zich zorgen moeten maken over onderhoud of kosten om deze servers te laten draaien. Dit stelt voor een eindgebruiker de grootste vrijheid van de 3 modellen. Daarnaast is er ook **PaaS** Waar de eindgebruiker, hetzij ontwikkelaars, hetzij zakelijke gebruikers, makkelijker en sneller applicaties kunnen ontwikkelen zonder enige zorg om het beheer van de servers. Als laatste is er ook **SaaS**. Bij SaaS wordt een software als online dienst geleverd aan de eindgebruiker (Hurwitz & Kirsch, 2020).

Een bedrijf is zelf verantwoordelijk voor de **secrets management**, het is anders ook niet logisch dat een service provider voor elke gebruiker de **secrets** gaat beheren. Soms passeert er wel de term 'Secrets as a Service', maar dit is geen populaire term. Hier is er een service van, genaamd **AWS IAM**, waar gebruikers zich zelf moeten laten identificeren voor authenticatie te verkrijgen om *secret services* te gebruiken, onder andere toegang tot bepaalde resources (thoughtworks, 2019). Dit is een tool van amazon web services (AWS) om cloud resources op een veilige manier te gebruiken en beheren. Hiermee de volgende vraag of er nog mogelijke cloud oplossingen zijn die een mogelijkheid aanbieden om **secrets** te beheren? Enkele mogelijkheden zijn:

- Cloud KMS van Google Cloud [4]
- Key Vault van Microsoft Azure [5]
- AWS Secrets Manager van Amazon Web Services [6]

Het is interessant om te zien welke applicaties en technieken er tegenwoordig bestaan om **secrets** zo goed mogelijk te beheren.

[4] https://cloud.google.com/security-key-management
[5] https://azure.microsoft.com/en-us/services/key-vault/
[6] https://aws.amazon.com/secrets-manager/

A.3 Methodologie

Om de onderzoeksvragen te kunnen beantwoorden zal er allereerst een literatuurstudie worden uitgevoerd om te weten welke benaderingen er bij **secret management** werden uitgevoerd en welke applicaties beschikbaar zijn om een *vault* aan te maken voor de **secrets** mee te beheren, verder wordt onderzocht bij welke applicaties er integratie mogelijkheden zijn met Teamcity [7]. Deze CI/CD applicatie maakt gebruik van *version control* om aan bepaalde bestanden te komen om applicaties op te bouwen, in deze bestanden eindigen soms belangrijke gegevens die geweerd moeten worden uit dit systeem. Er zal gekeken worden naar mogelijkheden om processen te automatiseren. Er wordt dan vooral gekeken naar een cloud oplossing en een on-premise opstelling. de verschillende tools worden opgezet en dienen het reële probleem op te lossen.

A.4 Verwachte resultaten

Er wordt verwacht dat via de implementaties van deze tools er duidelijk voorgesteld zal worden hoe **secret management** een belangrijk aspect is om op te nemen binnen een bedrijf en hoe deze voor een abstractielaag zorgt tussen **secrets** en taken die uitgevoerd worden met deze gegevens. Er wordt verwacht dat de opstelling niet te ingewikkeld gaat zijn om op te zetten en de interfaces gebruiksvriendelijk zijn om te gebruiken.

A.5 Verwachte conclusies

Uit dit onderzoek wordt er verwacht de conclusie te nemen hoe belangrijk het is voor bedrijven om op tijd te denken aan een vorm van **secrets management**. Men zal de mogelijkheid hebben om dit te waarnemen via een *proof of concept* en potentieel de mogelijkheid hebben om dit verder uit te breiden zodat het gebruik van **secrets** in een development omgeving, secret blijven.

[7] https://www.jetbrains.com/teamcity

Bibliografie

Azure, M. (2021). *Microsoft Azure*. Verkregen 27 mei 2021, van https://azure.microsoft.com/nl-nl/

Castle, B. (2019). What are the Cloud Deployment Models? Verkregen 16 april 2021, van https://www.youtube.com/watch?v=X1qoKLL040A&ab_channel=CBTNuggets

CodingPackets. (2019). Ansible. Verkregen 5 mei 2021, van https://codingpackets.com/code/ansible/

Confidant. (g.d.). Confidant. Verkregen 13 mei 2021, van https://lyft.github.io/confidant/contents.html

Dadgar, A. (2018). Introduction to HashiCorp Vault with Armon Dadgar. Verkregen 4 mei 2021, van https://www.youtube.com/watch?v=VYfl-DpZ5wM&ab_channel=HashiCorp

Digicert. (2021). *Digicert*. Verkregen 27 mei 2021, van https://www.digicert.com/

Duong, T. & Rizzo, J. (2011). Cryptography in the web: The case of cryptographic design flaws in asp. net. *2011 IEEE Symposium on Security and Privacy*, 481–489.

Ganatra, H. (2019). Securing data through Azure IAM and access control. Verkregen 6 mei 2021, van https://www.zensar.com/blogs/2019/03/securing-data-azure-iam-access-control/

GB, P. (2018). Protecting Your Secrets with Ansible. Verkregen 6 mei 2021, van https://medium.com/@pavithra_38952/protecting-your-secrets-with-ansible-cac601a1a5ab

Goyal, S. (2014). Public vs private vs hybrid vs community-cloud computing: a critical review. *International Journal of Computer Network and Information Security*, *6*(3), 20.

Hoeffnagel, W. (2020). Gartner: Investeringen in public cloud groeien komend jaar met achttien procent. Verkregen 3 januari 2021, van https://executive-people.nl/661926/

gartner-investeringen-in-public-cloud-groeien-komend-jaar-met-achttien-procent.html

Hoffman, B. (2021). *Can You Keep a Secret? Your Secrets Management System Can* (B. Hoffman, Red.). Verkregen 6 mei 2021, van https://thycotic.com/company/blog/2021/03/04/secrets-management/

Hurwitz, J. S. & Kirsch, D. (2020). *Cloud computing for dummies*. John Wiley & Sons.

Ingerson, B. (g.d.). Yet Another Markup Language (YAML) 1.0: Working Draft 01 Aug 2001. 2001.[Electronic resource]. *Mode of access: http://web.archive. org/web/20070217091403/http://yaml.org*, *80*.

Jiménez, J., Baig, R., Escrich, P., Khan, A. M., Freitag, F., Navarro, L., Pietrosemoli, E., Zennaro, M., Payberah, A. H. & Vlassov, V. (2013). Supporting cloud deployment in the Guifi. net community network. *Global Information Infrastructure Symposium-GIIS 2013*, 1–3.

Lugger, A. (2020). What is HashiCorp Vault and how does it work? Verkregen 7 mei 2021, van https://sensu.io/blog/what-is-hashicorp-vault-and-how-does-it-work

Lundberg, D. (2016). Open-sourcing Knox, a secret key management service. Verkregen 13 mei 2021, van https://medium.com/pinterest-engineering/open-sourcing-knox-a-secret-key-management-service-3ec3a47f5bb

Majumder, S. (2019). Azure Key Vault : Overview. Verkregen 8 mei 2021, van https://social.technet.microsoft.com/wiki/contents/articles/52480.azure-key-vault-overview.aspx

Malathi, M. (2011). Cloud computing concepts. *2011 3rd International Conference on Electronics Computer Technology*, *6*, 236–239.

Marczak, A. (2020). AZ-900 Episode 27 | Azure Key Vault | Secret, Key and Certificate Management. Verkregen 8 mei 2021, van https://www.youtube.com/watch?v=AA3yYg9Zq9w&ab_channel=AdamMarczak-AzureforEveryone

Mell, P., Grance, T. e.a. (2011). The NIST definition of cloud computing.

Microsoft. (2021). Wat is een privécloud? Verkregen 16 april 2021, van https://azure.microsoft.com/nl-nl/overview/what-is-a-private-cloud/

mizitechinfo. (2013). Cloud Computing : What You Need to Know? – Part 1 (The basic about Cloud Computing). Verkregen 16 april 2021, van https://mizitechinfo.wordpress.com/2013/09/12/cloud-computing-what-you-need-to-know-part-1-the-basic-about-cloud-computing/

oracle. (2021). Wat is SaaS? *https://www.oracle.com/be-nl/applications/what-is-saas/*.

Passi, H. (2018). What is a Sniffing attack and How can you defend it? Verkregen 6 mei 2021, van https://www.greycampus.com/blog/information-security/what-is-a-sniffing-attack-and-how-can-you-defend-it

Rayome, A. D. (2019). Ansible overtakes Chef and Puppet as the top cloud configuration management tool. Verkregen 5 mei 2021, van https://www.techrepublic.com/article/ansible-overtakes-chef-and-puppet-as-the-top-cloud-configuration-management-tool/

RedHat, I. (2021a). Encrypting content with Ansible Vault. Verkregen 5 mei 2021, van https://docs.ansible.com/ansible/latest/user_guide/vault.html

RedHat, I. (2021b). How Ansible Works. Verkregen 5 mei 2021, van https://www.ansible.com/overview/how-ansible-works

Sacic, B. (2020). What Is a Hybrid Cloud? Verkregen 23 april 2021, van https://itelligencegroup.com/cn/global-blog/what-is-a-hybrid-cloud/

Schoonen, D. (2021). 1Password koopt SecretHub en lanceert nieuwe secret management tool. Verkregen 13 mei 2021, van https://itdaily.be/nieuws/security/1password-koopt-secrethub-over-en-lanceert-nieuwe-secret-management-tool/

Shiner, J. (2021). 1Password koopt SecretHub en lanceert nieuwe secret management tool. Verkregen 13 mei 2021, van https://blog.1password.com/introducing-secrets-automation/

Somerfield, D. (2015). Daniel Somerfield - Turtles All the Way Down: Storing Secrets in the Cloud and the Data Center. Verkregen 4 mei 2021, van https://danielsomerfield.github.io/turtles/

Sullivan, E. (2015). pay-as-you-go cloud computing (PAYG cloud computing). Verkregen 9 april 2021, van https://searchstorage.techtarget.com/definition/pay-as-you-go-cloud-computing-PAYG-cloud-computing

TeamCity, J. (2021). *Jetbrains TeamCity*. Verkregen 27 mei 2021, van https://www.jetbrains.com/teamcity/

thoughtworks. (2019). *Secrets as a service*. Verkregen 6 mei 2021, van https://www.thoughtworks.com/radar/techniques/secrets-as-a-service

Tozzi, C. (2020). What is Secrets Sprawl How to Avoid It with Secrets Management. Verkregen 4 mei 2021, van https://www.conjur.org/blog/what-is-secrets-sprawl-how-to-avoid-it-with-secrets-management/

van Vugt, S. (g.d.). YAML (YAML Ain't Markup Language). Verkregen 5 mei 2021, van https://searchitoperations.techtarget.com/definition/YAML-YAML-Aint-Markup-Language

Vault, H. (2021). *Hashicorp Vault*. Verkregen 27 mei 2021, van https://www.vaultproject.io/

vyadh. (2018). Azure Key Vault TeamCity Plugin. Verkregen 27 mei 2021, van https://github.com/vyadh/teamcity-azure-keyvault-plugin

Wehner, J. (2015). How to undo (almost) anything with Git. Verkregen 5 mei 2021, van https://github.blog/2015-06-08-how-to-undo-almost-anything-with-git/

whitewaterfoundry. (g.d.). What is WSL? Verkregen 24 mei 2021, van https://www.whitewaterfoundry.com/what-is-wsl