Professionele Bachelor Toegepaste Informatica

Wachtwoordbeheer voor teams

Gijs Peerlings

Promotoren:

Peter Fastré — Level27
David Parren — Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Professionele Bachelor Toegepaste Informatica

Wachtwoordbeheer voor teams

Gijs Peerlings

Promotoren:

Peter Fastré — Level27
David Parren — Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Dankwoord

Deze scriptie heb ik geschreven tijdens de opleiding “Bachelor in de Toegepaste Informatica” aan de Hogeschool PXL. De opleiding heeft me veel geleerd over de informaticawereld.

Aan het begin van de scriptie wil ik graag een paar personen bedanken die hebben bijgedragen aan het tot stand komen ervan. Om te beginnen zou ik graag mijn stagebegeleider Peter Fastré willen bedanken. Zonder hem zou deze scriptie er niet zijn. Verder wil ik Roald Lenaerts bedanken voor zijn goede technische hulp. Ook gaat mijn dank uit naar alle medewerkers van Level27. Dankzij hun begeleiding is deze scriptie tot een goed einde gekomen.

Als laatste wil ik graag mijn ouders bedanken voor al hun hulp tijdens mijn studie. Zonder hen was ik nooit zo ver gekomen. Ook mijn vrienden bedank ik graag voor hun steun.

G. Peerlings

Abstract

In deze paper worden drie wachtwoordbeheerders met elkaar vergeleken. Een beheerder moet de volgende kwaliteiten bezitten: veiligheid, uitbreidingsmogelijkheden en de mogelijkheid om deze zelf te hosten. Dit onderzoek focust zich in de eerste plaats op de veiligheid en uitbreidingsmogelijkheden. Ook wordt bekeken of de wachtwoordbeheerder gebruiksvriendelijk is en hoe makkelijk het is om deze te installeren.

De beveiliging kijkt naar hoe de encryptie is opgebouwd en naar de verschillen in implementatie door de beheerders. Bij de uitbreidingsmogelijkheden wordt bekeken welke opties er zijn naast het bijhouden van het standaardwachtwoord. Wat gebruiksvriendelijkheid betreft wordt onderzocht hoe makkelijk de beheerder is in gebruik door niet-technisch personeel. Als laatste wordt bekeken hoe makkelijk het is om de manager te installeren en te repareren.

Het onderzoek omvat een literatuurstudie, in papers en online artikels is onderzocht hoe de wachtwoordbeheerder functioneert. Daarna wordt een *Proof of Concept* (PoC) onderzocht. Als laatste worden de resultaten samengebracht in een conclusie en wordt onderzocht welke wachtwoordbeheerder in welk onderdeel het sterkst is.

Inhoudsopgave

Lijst van gebruikte figuren

Lijst van gebruikte tabellen

Lijst van gebruikte afkortingen

PoC	Proof of Concept
KVM	Kernel-based Virtual Machine
SSH	Secure Shell
PKI	Public Key Infrastructure
RDP	Remote Desktop Protocol
SMS	Short Message Service
UI	User Interface
CLI	Command-Line Interface
API	Application Programming Interface
TLS	Transport Layer Security
AES	Advanced Encrypted Standard
NaCl	Networking and Cryptography Library
ECC	Elliptische Curve Cryptografie
ECDH	Elliptische Curve Diffie-Hellman
RSA	Rivest Shamir Adleman
VPN	Virtual Private Network
SSL	Secure Sockets Layer
PFS	Perfect Forward Secrecy/security
MAC	Message Authentication Code
ACL	Access-Control List
PGP	Pretty Good Privacy

Inleiding

Deze paper onderzoekt drie wachtwoordbeheerders op de gebieden veiligheid, uitbreidingsmogelijkheden, gebruiksgemak en configuratie. Ook wordt in de paper beschreven hoe de installatie en configuratie van deze beheerders verlopen.

In hoofdstuk 1 wordt de opdrachtgever voorgesteld en wordt uitgelegd waarom deze opdracht voor het bedrijf belangrijk is. Daarna wordt begonnen aan de uitwerking van de opdracht. Deze omvat o.a. de opzet van een werkende versie voor iedere beheerder en uitleg over vooraf benodigde kennis. Alle genomen stappen worden in beeld gebracht en daarna verklaard. Tot slot worden de uitbreidingen omschreven.

Hoofdstuk 2 omvat het onderzoek. Er wordt begonnen met de onderzoeksvraag en de deelvragen. Daarna wordt de onderzoeksmethode toegelicht, gevolgd door een korte inleiding. In het daaropvolgende deel worden de wachtwoordbeheerders apart onderzocht. Er wordt geprobeerd om een zo compleet mogelijk beeld te krijgen van hoe ieder deelprobleem door de wachtwoordbeheerders wordt opgelost. Ook worden per wachtwoordbeheerder de nadelen bekeken. Als laatste is er de conclusie. Hierin worden de deelvragen samengebracht en wordt op een rij gezet welke wachtwoordbeheerder het beste ieder probleem aanpakt. De beheerders worden hier met elkaar vergeleken.

I. Stageverslag

1 Bedrijfsvoorstelling

In dit hoofdstuk worden Level27 en het organigram van het bedrijf omschreven. Ook wordt beschreven wat het belang is van de stageopdracht en wat het nut daarvan is voor het bedrijf. Als laatste volgt de motivatie voor de keuze voor Level27.

1.1 Level27

Level27 is gespecialiseerd in de *hosting* van websites en het beheren van servers. Het bedrijf wil zich onder andere onderscheiden van andere bedrijven door goede *support* te bieden aan zijn klanten. Met de recente overname van Sitehosting en een steeds groeiend klantenbestand is Level27 een gezonde organisatie. De visie van Level27 is kort samengevat “de klant is koning” en “de klant heeft altijd gelijk”.

Figuur 1 - Logo Level27

Website hosting

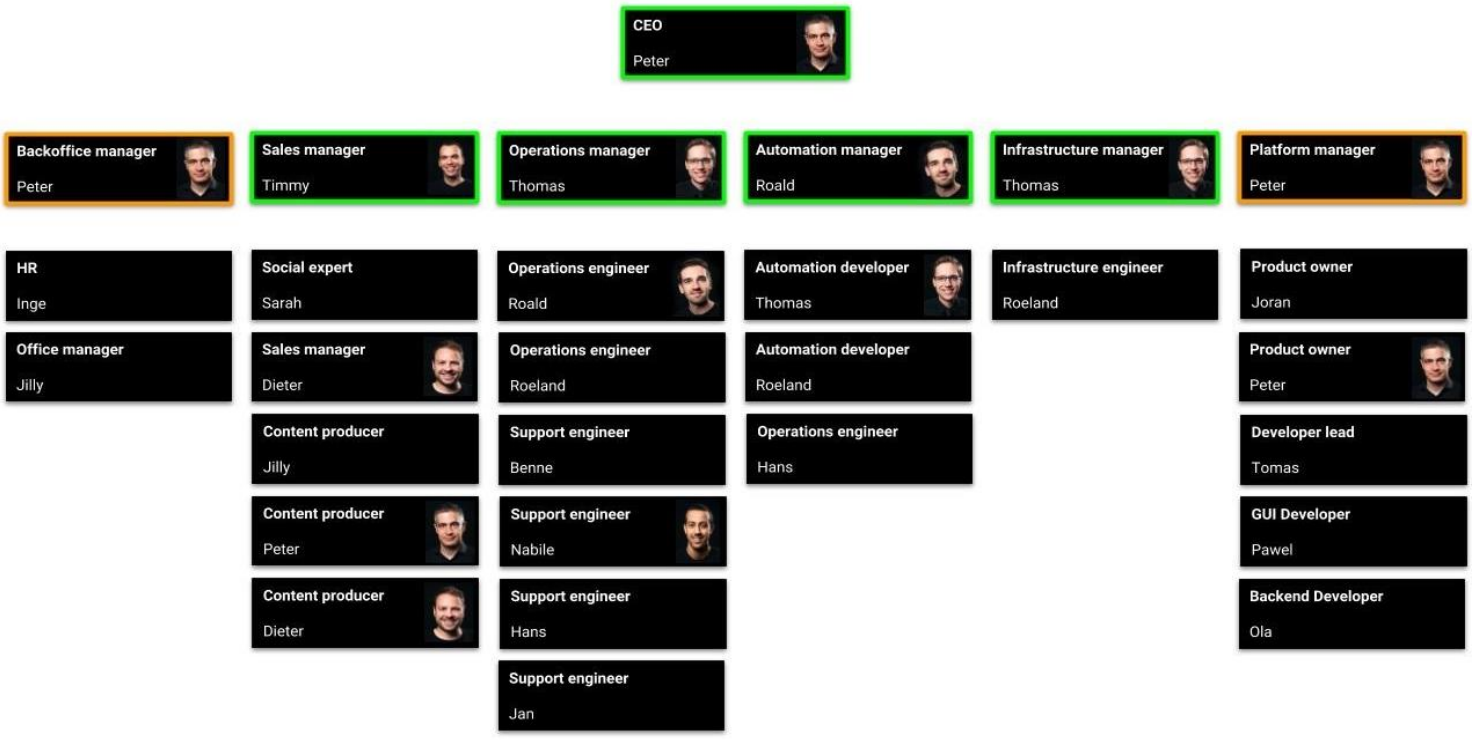
Onder *website hosting* vallen *shared hosting* en *dedicated hosting*. Het verschil tussen de twee is dat bij *shared hosting* meerdere websites op één machine staan en *dedicated hosting* één website per machine heeft. *Shared hosting* is dus meestal bedoeld voor websites met weinig webverkeer. Er is ook een optie om alleen een domein te kopen en alles zelf te *hosten*.

Serverbeheer

Level27 geeft ook de optie om een hele ICT-infrastructuur in beheer te nemen. Alle servers worden dan geïnstalleerd en beheerd door Level27. Klanten kunnen dit ook zelf doen als zij dit willen.

1.2 Bedrijfsorganigram

Figuur 2 geeft een overzicht van de medewerkers van Level27 en hun functie. Sommige medewerkers hebben meerdere functies en dus ook meer verantwoordelijkheden.



Figuur 2 - Organigram

1.3 De opdracht en Level27

Om klanten beter van dienst te kunnen zijn en de huidige processen sneller en makkelijker te laten verlopen, heeft Level27 een goede "*password manager*" nodig. Er wordt een stagiair ingezet om uit te zoeken wat de beste opties hiervoor zijn en welke uitbreidingen mogelijk zijn.

Na het voltooien van de opdracht moet een werkende *password manager* in gebruik zijn waar iedereen van het team makkelijk wachtwoorden mee kan opvragen.

Level27 stelt een omgeving ter beschikking voor het maken van een *Proof of Concept* (PoC), ook voor een definitieve versie. Deze omgeving is gebouwd met het eigen controlepaneel dat op de achtergrond een *Kernel-based Virtual Machine* (KVM) gebruikt. Wanneer nodig kan de hypervisor altijd worden aangepast.

1.4 Motivering

Een goede omgang met klanten is waar Level27 voor staat. Voor een stagiair is er veel te leren over verschillende systemen, van Windows tot Linux.

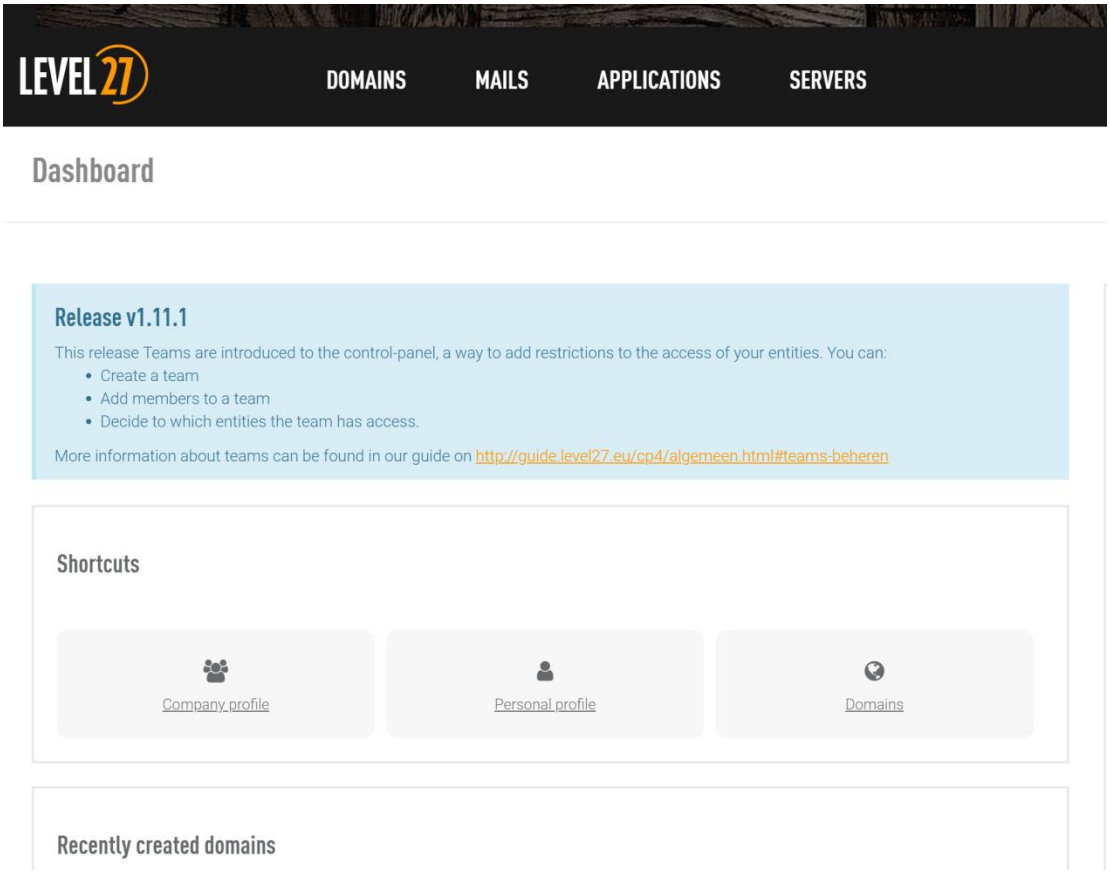
De keuze voor Level27 is gemaakt omdat het een kleiner bedrijf is, wat teamwork en dus samenwerken makkelijker maakt. Verder is het bedrijf goed aan het groeien en zijn er genoeg kansen om door te groeien naar hogere functies voor wanneer ik na de stage zou worden aangenomen.

1.5 Aanleiding opdracht

Om de productiviteit en efficiëntie van Level27 te verhogen, wil het bedrijf een *password manager* om alle sensitieve data bij te houden. In de toekomst kan deze gebruikt worden om niet alleen wachtwoorden bij te houden, maar ook *Secure Shell* (SSH)-*keys*, *Public Key Infrastructure* (PKI)-certificaten en *Remote Desktop Protocol* (RDP)-aanmeldingsgegevens. Op dit moment worden alle wachtwoorden bijgehouden op een *Confluence* Wikipagina. SSH-gegevens worden intern doorgestuurd via het programma Resiliosync.

1.6 Automatisatie

Level27 zet groot in op automatisatie. Dit doet het bedrijf met de programmeertaal Chef. Voor normale gebruikers is hier niets van zichtbaar, alles kan namelijk geconfigureerd worden in het controlepaneel. Het controlepaneel is volledig intern gebouwd door Level27. Om bijvoorbeeld de *Hypertext Preprocessor* (PHP) te installeren voldoen een paar muisklikken. Hieronder staat een afbeelding van het controlepaneel (figuur 3).



Figuur 3 - Controlepaneel Level27

1.7 Missie

De missie van Level27 is zoals gezegd om alle klanten op de best mogelijke manier te ondersteunen. Verder staat het bedrijf voor:

- Kwaliteit
- Communicatie
- *No bullshit*
- Competentie
- Klantvriendelijkheid
- Veiligheid
- Verantwoordelijkheid
- Proactiviteit

2 Uitwerking stageopdracht

2.1 Inleiding

In dit hoofdstuk wordt uitgelegd hoe de opdracht precies is uitgevoerd. De opdracht is een geschikte wachtwoordbeheerder te vinden door het opzetten van een PoC. Deze wordt daarna geïnstalleerd in een productieomgeving. Eerst wordt basisuitleg gegeven over de *tools* die gebruikt worden in dit project.

2.2 Uitbreiding

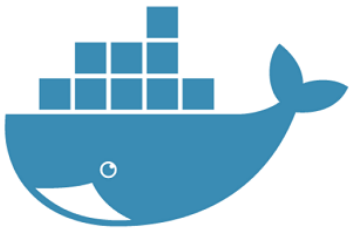
Zoals gezegd bestaat de basisopdracht erin om een werkende *password manager* tot stand te brengen. Indien er tijd over is, zijn de volgende uitbreidingen mogelijk:

- Clustering
- PKI-certificaten
- SSH *one-time password login*
- RDP-gegevens

2.3 Basisconcepten

2.3.1 Docker

Om het geheel makkelijk te kunnen automatiseren worden veel onderdelen in Docker gemaakt. De belangrijkste concepten van Docker worden hieronder verklaard.



Figuur 4 - Logo Docker [43]

2.3.1.1 Container [1]

Een container is een standaardpakket van software met alle afhankelijkheden van die software. Containers staan los van de rest van het systeem, waardoor problemen worden voorkomen. De bedoeling van Docker is om een zo klein mogelijke impact te hebben op het systeem. Containers zijn dan ook vaak ontdaan van alle onnodige software en bestanden.

2.3.1.2 Docker-compose [2]

Compose is een *tool* om een volledig systeem te beschrijven met behulp van een YAML-code. Op deze manier is het mogelijk om een volledig systeem met een commando klaar te zetten voor gebruik. *Compose* kan gebruikt worden in elk soort omgeving, van testen tot productie. *Compose* gebruiken is vaak een proces in drie stappen:

1. Het is belangrijk dat alle *images* goed gedefinieerd zijn, waarvoor eventueel een "Docker-*file*" gebruikt kan worden. In het volgende deel, *images*, wordt uitgelegd wat dit is.
2. Daarna kunnen alle *services* in een "docker-compose.yml"-bestand gezet worden, zodat ze samen in een afgesloten omgeving gelanceerd kunnen worden.
3. Ten slotte wordt het commando *"docker-compose up -d"* uitgevoerd om de applicaties te starten. '-d' staat voor *detached*.

Figuur 5 visualiseert een Docker-compose.yml-bestand.

```
version: '3'
services:
  web:
    build: .
    ports:
    - "5000:5000"
    volumes:
    - .:/code
    - logvolume01:/var/log
    links:
    - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

Figuur 5 - Voorbeeld Docker-compose.yml

2.3.1.3 Images [3]

Een Docker-*image* is een bestand dat bestaat uit meerdere lagen. Het wordt gebruikt om programma's uit te voeren in een container. Een *image* wordt gebouwd door gebruik te maken van een Docker-*file*. Deze bevat instructies voor een complete en uitvoerbare applicatie. Als Docker een *image* uitvoert, wordt dit een container.

Een Docker-*image* bestaat uit meerdere lagen. Alles wat wordt toegevoegd of verwijderd van een *image* voegt een nieuwe laag toe. Het is goed gebruik om de hoeveelheid lagen zo klein mogelijk te houden. De meeste Docker-*images* starten al vanaf een basis-*image*. Zo hoeft niet alles helemaal opnieuw te worden gemaakt. Deze *images* kunnen gevonden worden in een *repository*. Docker heeft zelf een *repository* genaamd Docker-Hub waar iedereen gratis *images* kan uploaden.

2.3.1.4 Volumes [3]

Als een nieuwe container van een *image* wordt gemaakt, komt hier een schrijfbare laag bovenop. Als de container dan wordt weggegooid, zijn deze data verdwenen. Volumes vormen een uitzondering hierop. Met een volume is het mogelijk data van buiten een container in een container te krijgen. De data worden dan niet verwijderd.

2.4 Technisch verloop

Voor deze opdracht voorziet Level27 een server. Het besturingssysteem is Ubuntu 18.04 LTS. Deze is geüpdatet tot de laatste versie om de beschikking te hebben over alle veiligheidsupdates.

2.4.1 Vault

De eerste *password manager* is Vault. Vault is een *password manager* van Hashicorp, het bedrijf dat o.a. Terraform [4] heeft gemaakt. Vault ondersteunt meer dan alleen wachtwoorden. Voorbeelden zijn SSH-*keys*, certificaten, Amazon Web *Services* (AWS)-logins, Google *cloud*gegevens en nog veel meer. Al deze gegevens worden opgeslagen in Consul. Ze zijn op te vragen via een *Web Userinterface* (UI), *Command-Line interface* (CLI) of *Application Programming interface* (API). Natuurlijk moet de gebruiker zichzelf wel eerst authenticeren.

Omdat Vault niet alleen klassieke wachtwoorden bijhoudt, wordt de term *secrets* in plaats van wachtwoorden gebruikt. In dit deel zullen daarom alle vormen van authenticatiemogelijkheden *secrets* genoemd worden.

2.4.1.1 Installatie

Om Vault te installeren zijn twee onderdelen nodig: een plaats om de *secrets* bij te houden en Vault zelf.

2.4.1.2 Opslag

De *secrets* moeten ergens worden opgeslagen. Vault kiest hiervoor Consul, een van de andere producten van Hashicorp. Beide programma's kunnen zeer makkelijk geclusterd worden en werken goed samen.

2.4.1.3 Consul

Consul is een *service mesh solution*. Een *service mesh* is bedoeld om service-naar-service-communicatie veilig, snel en makkelijk te maken. [5] Alle functionaliteiten van Consul kunnen zowel individueel of als geheel gebruikt worden. Hieronder worden de *features* van Consul kort toegelicht:

Service discovery: Consul-*clients* kunnen *services* als een API of mysql registreren. Andere Consul-*clients* kunnen deze dan makkelijker vinden.

Health checking: Consul-*clients* kunnen zoveel *Health checks* doen als gewenst is. Dit kan met een *service* of een lokale machine. De informatie die dit oplevert kan gebruikt worden om bijvoorbeeld te monitoren. *Service discovery* gebruikt deze informatie ook om *traffic* weg te leiden van slechte machines of *services*. Een voorbeeld van de informatie die de *health check* geeft is: "*The webserver is returning 200 OK*" of "*Memory utilization is below 90%*".

KV *Store*: Applicaties kunnen gebruikmaken van Consul's *key/value* opslag. Vault zal deze bijvoorbeeld gebruiken om *secrets* bij te houden. Er zijn echter veel meer mogelijkheden, bijvoorbeeld *leader election* en *feature flagging*. De KV Store is toegankelijk via een API.

Secure service Communication: Consul kan *Transport Layer Security* (TLS)-certificaten genereren en distribueren om *services* voor TLS-connecties op te zetten. Het kan aangeven welke *services* met elkaar mogen communiceren en welke niet. Dit kan door gebruik te maken van '*Intentions*'.

Multi Datacenter: Consul is gebouwd voor meerdere datacenters. Gebruikers hoeven zich dus geen zorgen te maken als ze uitbreiden naar nieuwe regio's. [6]

2.4.1.4 Installatie Consul [7]

De makkelijkste manier om Consul te installeren is door dit te doen via Docker. Hieronder staat het docker-compose.yml-bestand dat gebruikt wordt.

```
version: "2.0"

services:

 server:
  image: consul
  volumes:
   -  /consul:/consul/data
  command: agent -server -bind=194.32.154.14 -bootstrap-expect=2 -retry-join=185.3.216.82 -retry-join=185.3.216.84
  restart: always
  network_mode: host

volumes:
  consul:
```

Figuur 6 - Docker-compose Consul

De opties in dit bestand doen het volgende:

Image: dit geeft aan welke *image* gebruikt moet worden als basis. Als deze niet lokaal bestaat, wordt geprobeerd deze van de Docker-hub te downloaden.

Volume: een volume wordt gebruikt om de data in de aangegeven map ook buiten de Docker-container op te slaan. Als de container wordt verwijderd bestaat deze dus nog.

Command: dit is het commando dat wordt uitgevoerd in de container. -server wil zeggen: om een Consul-server te starten. -bind is het IP-adres dat Consul kan gebruiken. *-bootstrap-expect*=2 wil zeggen dat er twee andere Consul-machines in het cluster moeten komen. Dit wordt verder toegelicht in 'uitbreiding 1 – redundantie'. *-retry-join* probeert om met een Consul-cluster te verbinden.

Restart: bij een error herstart de container automatisch.

Network_mode: als deze optie op '*host*' staat, wordt er geen apart netwerk gecreëerd voor de Docker-container en gebruikt de Docker-container het netwerk van de *host*. Meestal is dit 'eth0'.

Om de container te starten bestaat het commando *docker-compose -d*. -d staat voor *detached* en zorgt ervoor dat de container op de achtergrond wordt gestart.

2.4.1.5 Installatie Vault [7]

Om de installatie af te maken is het nodig dat Vault zelf geïnstalleerd wordt. Op een Ubuntu 18.04-systeem kan dit door middel van een 'systemd'-service. Eerst dient Vault gedownload te worden. Dit kan met het commando 'wget'. De link naar de laatste versie is te vinden op https://www.vaultproject.io/downloads.html. Het gedownloade bestand is van het zipformaat en kan uitgepakt worden met de CLI-*tool* '*unzip*'. Om het helemaal af te maken kan het uitgepakte bestand het best ook opgeslagen worden onder '/usr/bin/vault'.

Een systemd-service wordt gemaakt door het bestand '/etc/systemd/system/vault.service' te maken. De inhoud hiervan wordt weergegeven in figuur 7.

```
[Unit]
Description=Vault
Documentation=https://www.vault.io/

[Service]
ExecStart=/usr/bin/vault server -config=/etc/vault/config.hcl
ExecReload=/bin/kill -HUP $MAINPID
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
```

Figuur 7 - Vault systemd-service

Om dit te laten werken is ook een configuratiebestand nodig. Dit staat in '/etc/vault/cofig.hcl'. De inhoud hiervan is te vinden in figuur 8.

```
storage "consul" {
  address = "127.0.0.1:8500"
  path    = "vault/"
}

listener "tcp" {
 address     = "0.0.0.0:8200"
 tls_disable = 0
 tls_cert_file = "/etc/letsencrypt/live/vault.level27.eu/fullchain.pem"
 tls_key_file = "/etc/letsencrypt/live/vault.level27.eu/privkey.pem"
}

ui = true
```

Figuur 8 - Vault configuratie

Zoals in figuur 8 te zien is, is er ook een letsencrypt-certificaat geïnstalleerd. Dit certificaat kan aangevraagd worden met de onderstaande commando's:

sudo add-apt-repository ppa:certbot/certbot
sudo apt install python-certbot-apache
sudo certbot certonly --agree-tos --email admin@example.com --webroot -w /var/lib/letsencrypt/ -d example.com -d www.example.com

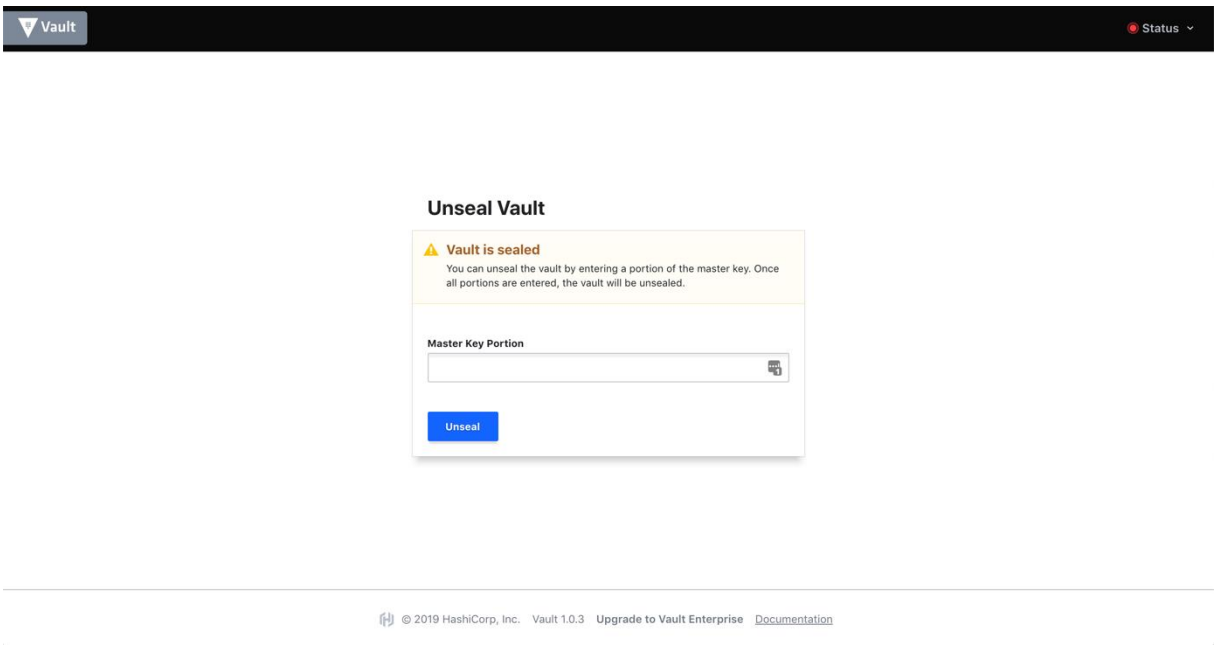
Example.com kan uiteraard vervangen worden door het gewenste domein.

De service kan gestart worden met het commando *'service vault start'*. Om de service samen te starten met het systeem kan deze ook op '*enabled*' worden gezet. Dit kan door het volgende commando: *'systemctl enable vault'*. Om Vault vervolgens te initialiseren kan het commando *'Vault init'* worden gebruikt. Dit stelt Vault in voor het eerste gebruik. Het genereert vijf sleutels, ook '*master keys*' genoemd. In het onderzoek worden deze sleutels verder toegelicht. Ook is er een '*root token*' gemaakt. Al deze toegangsgegevens dienen veilig bewaard te worden, eventueel in een kluis.

2.4.1.6 Configuratie [8]

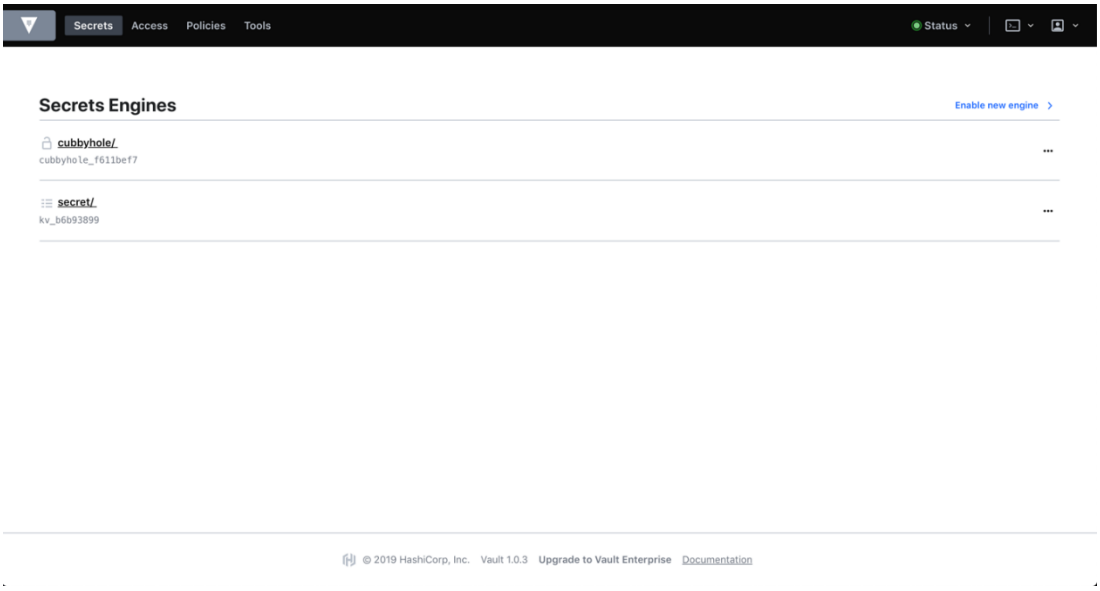
Om Vault in te stellen zijn er meerdere opties. Het makkelijkst is om dit via de webbrowser te doen, het is ook mogelijk om de CLI of de API te gebruiken.

Als eerste moet een vorm van authenticatie gebruikt worden die geen gebruik maakt van de *root token*. Als er wordt genavigeerd naar het IP-adres van de machine op poort 8200, is de *homepage* van Vault zichtbaar. Dit wordt gevisualiseerd in figuur 9.



Figuur 9 - Vault unseal pagina

Hier moeten drie van de vijf *master keys* worden ingegeven om de Vault te openen. Deze werden gemaakt in hoofdstuk 3.4.1.5. Als er op '*unseal*' wordt geklikt na het invoeren van de drie *master keys*, wordt Vault geopend en is het mogelijk om in te loggen met de *root token*. Daarna wordt het hoofdscherm van Vault weergeven (Figuur 10).



Figuur 10 - Homepage Vault

De *secret engines* zijn de vormen van wachtwoorden die worden bijgehouden. De KV-*engine* wordt bijvoorbeeld gebruikt om gebruikersnamen met wachtwoorden bij te houden. KV staat voor *Key/Value*. Andere vormen van *engines* zijn de SSH-*engine* en PKI-*engine*.

Om een nieuwe gebruiker te installeren moeten eerst de rechten voor die gebruiker bepaald worden. Dit is belangrijk omdat hoogstwaarschijnlijk niet iedere gebruiker alle opties mag configureren en alle wachtwoorden mag zien. Om een nieuw gebruikersbeleid te maken, wordt er naar '*Policies*' genavigeerd en daarna geklikt op '*Create ACL policy*'. Een naam kan zelf gekozen worden. Het beleid dat gemaakt wordt is een '*Admin policy*' met iets minder rechten dan de *root token*. In bijlage A zit het hele beleid. In het volgende deel staat meer uitleg over ACL.

Op Level27 is bepaald dat er wordt inlogt in Vault met een gebruikersnaam en wachtwoord. Tijdelijke medewerkers krijgen een '*token*'. De eerste methode kan ingesteld worden bij *Access -> Auth Methods -> Enable new method -> Username & Password -> Enable*. Er moeten dan nog gebruikers worden aangemaakt. Op dit moment is iedere medewerker van Level27 een administrator. Dit wordt in het deel over ACL aangepast.

Om een nieuwe gebruiker aan te maken kan er rechtsboven een CLI in de webpagina geopend worden. Om de gebruiker 'bob' te maken wordt het volgende commando gebruikt.

```
vault write auth/userpass/users/bob password="training" policies="Admin"
```

Het wachtwoord en de *policy* moeten natuurlijk vervangen worden. Als dan op '*Entities*' geklikt wordt, kan een nieuwe '*Entity*' worden aangemaakt met een zelfgekozen naam. Als alle gegevens zijn ingevuld kan deze worden aangemaakt en daarna aangepast. Om deze *entity* te koppelen aan de login wordt '*Add alias*' geselecteerd en dan de naam 'bob' gekozen. Er kan nu ingelogd worden als gebruiker bob.

Vervolgens kunnen *secrets* aangemaakt worden en verwijderd in Vault. Deze kunnen door alle gebruikers met toegang worden ingezien en aangepast. De toegang wordt zoals eerder genoemd gegeven in een beleid of ACL.

2.4.1.7 Beleid en ACL

Onder '*policies*' kunnen de rechten voor iedere gebruiker of voor een groep gebruikers worden ingesteld. Dit gebeurt met een ACL. Een ACL is een configuratiebestand dat aanduidt welke rechten een gebruiker heeft op een bepaald pad. Alle instellingen in Vault zijn een pad. In het vorige deel is de authenticatiemogelijkheid "*Username & Password*" via de UI aangezet. Het pad waar dit aan gelijkstaat is: '*auth/userpass/**'. Bij Level27 zijn er drie '*policies*' gemaakt. Deze worden in tabel 1 weergeven. [9] Hier is ervoor gekozen om de administrator ook toe te staan om zijn eigen rechten te beheren. Als een snelle aanpassing nodig is, is het namelijk niet praktisch om hiervoor de *root token* uit de kluis te halen. De *root token* is tijdens de installatie aangemaakt.

Tabel 1 - Overzicht ACL-rechten

	Administrator	Operations	(Job)student
Policies beheren	X		
Toegang beheren	X		
Nieuwe '*secret' engine* aanmaken	X		
Wachtwoorden maken	X	X	
Wachtwoorden veranderen	X	X	
Alle wachtwoorden lezen	X	X	
Specifieke selectie wachtwoorden lezen	X	X	X

2.4.2 Passbolt

De tweede *password manager* is Passbolt. Dit is een vrij nieuwe *password manager* die wordt gemaakt in Luxemburg. Passbolt is nog volop in ontwikkeling, maar kan al gebruikt worden om geheimen in op te slaan. In het komende jaar worden veel functionaliteiten toegevoegd, waaronder ondersteuning voor PKI-infrastructuur.

2.4.2.1 Installatie

Passbolt heeft evenmin als Psono een aparte *storage* nodig om te functioneren. Wat wel nodig is, is een database. Er kan zelf gekozen worden welke gebruikt wordt. In deze scriptie wordt mariadb gebruikt, maar deze kan indien gewenst gewisseld worden voor een andere databaseoplossing.

2.4.2.2 Mariadb in Docker

Om mariadb te installeren in Docker kan het volgende commando worden gebruikt:

```
docker run --name mariadb -e MYSQL_ROOT_PASSWORD=password \
    -e MYSQL_DATABASE=passbolt \
    -v /var/lib/mysql:/var/lib/mysql \
    -e MYSQL_USER=rootie \
    -e MYSQL_PASSWORD=password \
    -d mariadb
```

Dit doet het volgende:

- --*name*: zet de naam voor Docker op mariadb. Dit is nodig om deze makkelijker te identificeren.
- -e: deze opties zetten *environment variables* in de Docker-container, zodat mariadb automatisch geïnstalleerd kan worden.
- -v: dit zorgt ervoor dat de database niet verwijderd wordt als de container verwijderd wordt. De data van de database staan dan nog op het pad /var/lib/mysql.
- -d: zorgt dat de container op de achtergrond wordt uitgevoerd.
- Mariadb: geeft aan welke *image* gebruikt moet worden. Om er zeker van te zijn dat het gaat om de nieuwste versie van mariadb, kan ook *mariadb:latest* gebruikt worden.

2.4.2.3 Installatie Passbolt

Passbolt wordt eveneens met Docker geïnstalleerd. Dit gebeurt door het volgende commando:

```
docker run --name passbolt --link mariadb:mariadb \
        -p 8080:80 \
        -p 444:443 \
        -e DATASOURCES_DEFAULT_HOST=mariadb \
        -e DATASOURCES_DEFAULT_PASSWORD=password \
        -e DATASOURCES_DEFAULT_USERNAME=rootie \
        -e DATASOURCES_DEFAULT_DATABASE=passbolt \
        -e APP_FULL_BASE_URL=https://passbolt.d2b627d2f.firecontrol.be \
        -d passbolt/passbolt:latest
```

De opties worden hieronder verder toegelicht:

- -p: de poorten waarop Passbolt beschikbaar wordt. Het eerste getal voor de ':' is de externe poort en het getal na de ':' is de poort waarop de applicatie beschikbaar is in de Docker-container.
- -e: APP_FULL_BASE_URL: dit is de URL waarop Passbolt beschikbaar wordt. Passbolt gebruikt dit om zichzelf te configureren.

Omdat er meerdere services op poort 80 en 443 gehost worden, is een nginx *reverse proxy* nodig. De configuratie hiervoor is terug te vinden in Bijlage B. Deze omvat ook de configuratie voor Psono, de derde *password manager*. Het is voor Psono vereist om een nginx *reverse proxy* te gebruiken. Nginx kan geïnstalleerd worden met het commando *apt install nginx*. Daarna kan de configuratie in */etc/nginx/sites-enable/jouwconf.conf* geplaatst worden.

2.4.2.4 Configuratie Passbolt

Zodra nginx klaarstaat, kan Passbolt opgezocht worden via passbolt.d2b627d2f.firecontrol.be. De URL is natuurlijk anders als deze is veranderd in nginx. Voordat deze veilig bereikbaar is, dient ook een letsencrypt-certificaat aangevraagd te worden. Indien de configuratie van Bijlage B wordt gebruikt, voldoen de volgende commando's: [10]

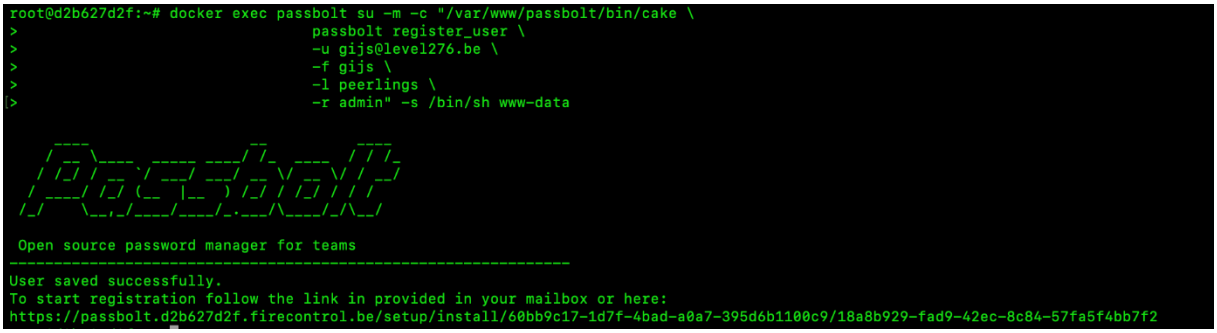
```
sudo add-apt-repository ppa:certbot/certbot
sudo apt install python-certbot-nginx
sudo certbot --nginx -d d2b627d2f.firecontrol.be -d passbolt.d2b627d2f.firecontrol.be
```

Om Passbolt te gebruiken is de extensie van het bedrijf nodig. Deze is met de onderstaande link te downloaden als Google Chrome gebruikt wordt:
https://chrome.google.com/webstore/detail/passbolt-extension/didegimhafipceonhjepacocaffmoppf?hl=en.

Op de server dient eerst een eerste gebruiker te worden aangemaakt. Dit gaat met het volgende commando:

*docker exec passbolt su -m -c "/var/www/passbolt/bin/cake *
*passbolt register_user *
*-u gijs@level27.be *
*-f gijs *
*-l peerlings *
-r admin" -s /bin/sh www-data

Er komt dan een *prompt* gelijkaardig aan figuur 11:



Figuur 11 - Passbolt installatie

Volg dan de getoonde link om de registratie te voltooien. Er kunnen nu ook andere gebruikers aangemaakt worden. Geheimen kunnen ook bijgehouden worden door in te loggen op de geconfigureerde *hostname*.

2.4.3 Psono

De laatste *password manager* is Psono. Dit is een redelijk onbekende manager die werkt op dezelfde manier als Passbolt. Psono wordt nog steeds geüpdatet, maar is anders dan Passbolt niet nog in bèta. Psono wil transparant en veilig zijn. Om Psono te installeren is Docker vereist; indien er drie virtuele machines beschikbaar zijn hoeft dit echter niet.

2.4.3.1 Installatie

Om Psono te installeren is ook Postgres vereist. Psono gebruikt functies in Postgress die niet beschikbaar zijn in andere oplossingen. Dit zijn de extensies ‘ltree’ en ‘pgcrypto’. Er wordt ook weer van Docker gebruikgemaakt voor de installatie. Psono omvat in tegenstelling tot Vault en Passbolt vier onderdelen. Dit zijn:

- Psono client UI
- Psono admin UI
- Psono server
- Postgress

Deze dienen allemaal apart geïnstalleerd te worden en vervolgens aan elkaar gelinkt.

2.4.3.2 Postgress in Docker [11]

Postgress kan in Docker gestart worden met het volgende commando:

```
docker run --name psono-database \
 -v /opt/docker/psono/postgres:/var/lib/postgresql/data \
 -e POSTGRES_USER=psono \
 -e POSTGRES_PASSWORD=password \
 -d --restart=unless-stopped \
 -p 5432:5432 postgres:9-alpine
```

De opties werken hetzelfde als bij de mariaDB van Passbolt. Als toevoeging is hier *–restart=unless-stopped* meegegeven. Dit doet precies wat er staat: de container herstarten als de applicatie uitvalt. Alleen als de container door een administrator gestopt wordt stopt deze.

2.4.3.3 Installatie Psono

Om te beginnen zijn geheime codes nodig. Deze kunnen gegenereerd worden met het volgende commando:

```
docker run --rm -ti psono/psono-server:latest python3 ./psono/manage.py generateserverkeys
```

In bijlage C is een configuratiebestand toegevoegd. Dit moet in /opt/docker/psono/settings.yaml geplaatst worden. Dit bestand dient aangevuld te worden met de gegenereerde gegevens en de databasegegevens.

Ook moet volgend bestand aangemaakt worden in /opt/docker/psono-client/config.json:

```
{
  "backend_servers": [{
    "title": "Psono client interface",
    "url": "https://www.d2b627d2f.firecontrol.be/server",
    "domain": "level27.be"
  }],
  "base_url": "https://www.d2b627d2f.firecontrol.be/",
  "allow_custom_server": false
}
```

Als deze bestanden klaar zijn, kan een volgend commando uitgevoerd worden om de database klaar te maken voor gebruik:

```
docker run --rm   -v /opt/docker/psono/settings.yaml:/root/.psono_server/settings.yaml -ti --link psono-
database:psono-database psono/psono-server:latest python3 ./psono/manage.py
migrate
```

Als laatste dienen de andere drie containers gestart te worden. Dit kan met de volgende commando's:

```
docker run --name psono-server \
  --sysctl net.core.somaxconn=65535  \
  --link psono-database:psono-database \
  -v /opt/docker/psono/settings.yaml:/root/.psono_server/settings.yaml \
  -d --restart=unless-stopped -p 10100:80 psono/psono-server:latest [12]

docker run --name psono-client \
  -v /opt/docker/psono-client/config.json:/usr/share/nginx/html/config.json \
  -d --restart=unless-stopped -p 10101:80 psono/psono-client:latest [13]

docker run --name psono-admin-client \
  -d --restart=unless-stopped -p 10102:80 psono/psono-admin-client:latest [14]
```

Zoals getoond in deel 3.4.2 is het belangrijk dat een nginx reverse proxy is geïnstalleerd. Als dit het geval is, is Psono bereikbaar op de *hostname* van de machine.

2.5 Uitbreiding 1: Redundantie

Als eerste uitbreiding wordt de *storage* van Vault (Consul) redundant gemaakt. In deel 2.4.1.5 worden tijdens de installatie al wat opties meegegeven. Dit zijn de opties *–retry-join* en *--bootstrap-expect*. Beide opties worden ook in deel 2.4.1.5 kort toegelicht. Om aan de '*quorum size*' te voldoen moeten drie servers met Consul worden gemaakt. Dit geeft een fouttolerantie van één server. Zolang dus twee van de drie servers werken, zijn de data van Vault veilig. De opdrachtgever hecht er geen belang aan om Vault ook redundant te maken. De belangrijke data worden immers in Consul opgeslagen.

Om de redundantie te laten werken is *bootstrap-expect* op 2 gezet, dit om ervoor te zorgen dat er op elk moment twee andere Consul-servers verwacht worden. *Retry-join* is bij elke server twee keer meegegeven om de andere twee servers mee te geven. Deze optie is dus uniek voor iedere Consul-server. [15]

2.6 Uitbreiding 2: SSH

De tweede uitbreiding voegt functionaliteit toe voor *one time* SSH-*passwords*. Op deze manier kan er voor een server een eenmalig wachtwoord gegenereerd worden. Om te beginnen moet de 'Vault-SSH-helper' geïnstalleerd worden. Deze kan worden gevonden op https://releases.hashicorp.com/vault-ssh-helper/. Deze kan worden gedownload, het uitgepakte bestand kan dan worden geplaatst op het volgende pad: /usr/local/bin/vault-ssh-helper.

De volgende commando's kunnen worden uitgevoerd om /etc/pam.d/sshd te configureren:

```
export SSHD_CONFIG_PATH=/etc/ssh/sshd_config
export PAMD_CONFIG_PATH=/etc/pam.d/sshd # enable ChallengeResponseAuthentication
# disable common-auth
sed -i -e 's/^@include common-auth/#@include common-auth/g' ${PAMD_CONFIG_PATH}
# allow Helper to use pam_exec
echo "auth requisite pam_exec.so quiet expose_authtok log=/tmp/vaultssh.log /usr/local/bin/vault-ssh-
helper -config=/etc/vault-helper.d/config.hcl" | tee -a ${PAMD_CONFIG_PATH}
echo "auth optional pam_unix.so not_set_pass use_first_pass nodelay" | tee -a ${PAMD_CONFIG_PATH}
```

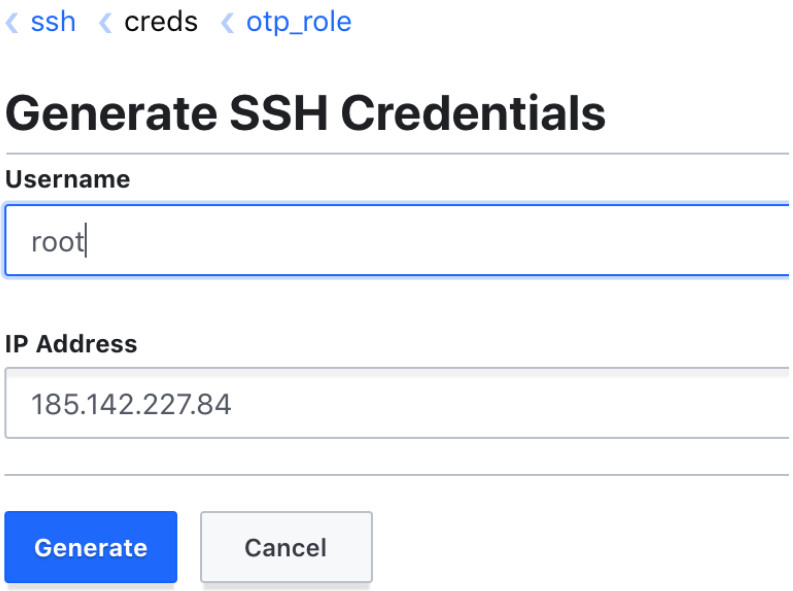
Daarna dienen de volgende commando's nog uitgevoerd worden om de configuratie van SSH zelf in orde te brengen:

```
sed -i -e 's/ChallengeResponseAuthentication no/ChallengeResponseAuthentication yes/g'
${SSHD_CONFIG_PATH}
# allow to use PAM
sed -i -e 's/UsePAM no/UsePAM yes/g' ${SSHD_CONFIG_PATH}
# disable password authentication
sed -i -e 's/PasswordAuthentication yes/PasswordAuthentication no/g' ${SSHD_CONFIG_PATH}
# restart SSH server
systemctl restart sshd
```

Als laatste moet /etc/vault.helper.d/config.hcl worden aangemaakt:

```
mkdir /etc/vault-helper.d/
# generate config for helper
cat << EOF > /etc/vault-helper.d/config.hcl
vault_addr = "https://vault.yourdomain.org"
ssh_mount_point = "ssh"
tls_skip_verify = false
allowed_roles = "*"
allowed_cidr_list="0.0.0.0/0"
EOF
```

Als er nu als Administrator wordt ingelogd op Vault en de SSH-*secret engine* wordt aangezet, kan een OTP gegeneerd worden. Dit wordt weergeven in figuur 12. [16]



Figuur 12 - SSH OTP genereren

3 Reflectie

De stage bij Level27 was zeer leerzaam. Met name heb ik leren werken in teamverband, ook heb ik op technisch gebied veel over de systemen van Level27 geleerd. Deze systemen hebben mij een andere kijk gegeven op wat ik op school heb geleerd. Van dingen waar ik voorheen het nut niet van inzag, zie ik dat nu wel.

Aan het begin van de stage heb ik voornamelijk aan het onderzoek en dit document gewerkt. Daarna heb ik meegewerkt aan projecten van Level27. De stage is vlot verlopen en ik heb geen grote problemen ondervonden. Iets wat ik wel beter kan doen is communiceren. Dit wil ik dan ook zeker doen.

Ook heb ik technologieën ontdekt die ik nog niet kende. Een voorbeeld hiervan is Varnish. Als laatste heb ik ondervonden hoe het is om echt voor een IT-bedrijf te werken.

II. Onderzoekstopic

Het volgende deel behandelt de onderzoeksopdracht van de stage. In het eerste hoofdstuk wordt de onderzoeksvraag verwoord. Vervolgens wordt toegelicht welke methoden gebruikt zijn bij de uitvoering van het onderzoek. Als laatste worden de resultaten van het onderzoek besproken en wordt daaruit een conclusie getrokken.

1 Onderzoeksvraag

Om met meer mensen te kunnen werken aan projecten met veel wachtwoorden, houden veel bedrijven deze wachtwoorden vaak gewoon zonder encryptie bij. Op deze manier zijn ze snel en makkelijk bereikbaar, maar komt veiligheid op de tweede plaats. Het is echter ook mogelijk om wachtwoorden veilig en zelfs nog makkelijker bij te houden. Dit kan door gebruik te maken van wachtwoordbeheerders oftewel *password managers*. In de uitwerking van de stageopdracht zijn drie wachtwoordbeheerders geconfigureerd, namelijk: Vault, Psono en Passbolt. Er zijn een paar te beantwoorden vragen:

- Welke veiligheidsmaatregelingen zijn ingebouwd en hoe werken deze precies?
- Welke wachtwoordbeheerder is het makkelijkst in gebruik?
- Wie heeft de meeste functionaliteit?
- Wat is het makkelijkst op te zetten?

2 Onderzoeksmethode

Het doel van de stageopdracht is om de opties van de verschillende wachtwoordbeheerders tegen elkaar af te wegen, wat leidt tot de beste keuze. De beste keuze moet veilig zijn en ook genoeg uitbreidingsmogelijkheden hebben. Onder uitbreidingen wordt bijvoorbeeld verstaan dat er *private keys* kunnen worden bijgehouden. De wachtwoordbeheerder moet ook redundant zijn en makkelijk te automatiseren, zodat in geval van nood de data toch toegankelijk blijven.

Het onderzoek maakt een vergelijking tussen drie *tools*. Eerst wordt een literatuurstudie uitgevoerd waarin gekeken wordt naar de bestaande documentatie. Daarna worden de toepassingen getest. Op deze manier wordt geprobeerd een goed beeld te krijgen van ieder onderdeel van de applicatie.

Het onderzoek is tot een goed einde gekomen als alle deelvragen in de onderzoeksvraag zijn beantwoord voor alle drie de toepassingen.

In dit onderzoek worden vooral “*self hosted password managers*” bekeken. Het doel van dit onderzoek is dat een lezer na het lezen van dit onderzoeksrapport direct weet welke “*self hosted password manager*” de beste keuze is.

3 Uitwerking onderzoek

3.1 Inleiding

In deze inleiding wordt uitgelegd wat een *password manager* precies is en welke soorten er bestaan. Het is belangrijk om dit te weten om een juiste keuze tussen de verschillende mogelijkheden te kunnen maken.

3.1.1 Wat zijn wachtwoordbeheerders? [17]

Wachtwoordbeheerders proberen in eerste instantie geheime informatie veilig op te slaan. Het tweede doel is deze geheime informatie op een makkelijke manier te kunnen delen met de gebruiker. Aangezien veilig en makkelijk lang niet altijd hand in hand gaan, is dit vaak een uitdaging. Er zijn dan ook veel verschillende manieren waarop dit probleem wordt aangepakt.

3.1.2 Soorten wachtwoordbeheerders

Er zijn drie verschillende soorten *password managers*. In dit onderdeel worden deze verder toegelicht.

3.1.2.1 Particuliere software [17]

De meeste computergebruikers hebben vaak al een *password manager* geïnstalleerd in hun browser. Dit zijn *password managers* die zijn gericht op het bijhouden van wachtwoorden van eindgebruikers of individuen. Hier vallen ook browserextensies en andere computerprogramma's onder die hetzelfde doen.

3.1.2.2 Cloudmanagers [17]

Verder zijn er wachtwoordbeheerders voor teams of organisaties. Een groep hiervan slaat deze wachtwoorden op in de *cloud*. Dit zorgt ervoor dat het beheer van de veiligheid en applicatie uit handen wordt genomen. Het nadeel is dat er hierdoor geen of weinig zicht is op wat er precies met de data gebeurt en hoe deze worden opgeslagen. Het voordeel is dat ze vaak makkelijker zijn ìn gebruik.

3.1.2.3 Self hosted managers

Als laatste zijn er de *self hosted password managers*. Zoals de naam al suggereert worden deze door organisaties of individuen zelf geïnstalleerd en beheerd. Dit zorgt ervoor dat de veiligheid volledig in eigen beheer is. Een nadeel van deze optie is dat de installatie en configuratie moeilijker is dan bij *cloudmanagers*.

3.1.3 Toekomst van wachtwoordbeheer [18]

Wachtwoorden worden nu al vaak gezien als onveilig. Er wordt daarom aangeraden om niet alleen een wachtwoord te gebruiken. Veel websites voorzien bijvoorbeeld al in "2-factor authenticatie". Er zijn verschillende manieren om dit te implementeren, bijvoorbeeld via *short message service* (SMS). "2-factor authenticatie" is jammer genoeg ook al niet meer zo veilig als deze fout geïmplementeerd is, wat vaak het geval is. [19]

Er zijn ook veiligere manieren. Zo is er het PKI-systeem, dat werkt met *private* en *public keys* om iemand te authenticeren. Ook is er biometrische authenticatie. Dit is al zichtbaar in smartphones, waar bijvoorbeeld vingerafdrukken als wachtwoord kunnen dienen.

3.2 Vault

3.2.1 Functies en uitbreidingen

Password managers bevatten verschillende functionaliteiten. Hieronder worden die van Vault bekeken. Vault omvat ondersteuning voor veel verschillende platformen. Deze worden gevisualiseerd in tabel 2.

Tabel 2 - Secret engines Vault

Generic	Cloud	Infrastructure
Key/Value	Active Directory	Consul
PKI Certificates	AliCloud	Databases
SSH	AWS	Nomad
Transit	Azure	RabbitMQ
TOTP	Google Cloud	
	Google Cloud KMS	

Al deze '*secret engines*' kunnen toegangsgegevens bijhouden voor de gelijknamige *service*. Deze kunnen door de gebruiker op verschillende manieren opgevraagd worden. Sommige *engines* houden gewoon data bij, andere verbinden naar andere services om bijvoorbeeld een eenmalig wachtwoord te genereren. [20]

Buiten deze functionaliteiten is het mogelijk om Vault te clusteren. Deze mogelijkheid is ingebouwd in de applicatie. Om redundant te zijn plaatst een Vault-server een slot in de opslag. Alleen deze Vault-server heeft dan toegang tot de opslag, de andere Vault-servers in een cluster gaan in stand-by. Als een aanvraag bij een stand-by-server aankomt, wordt deze doorgestuurd of wordt de gebruiker doorverwezen. Deze architectuur is dus niet schaalbaar. De *bottleneck* is echter vaak de opslag. Deze kan vaak makkelijker geschaald worden, in het geval van Vault gebeurt dit door Consul te schalen. [21]

3.2.2 Beveiliging

Password managers moeten een goede beveiliging hebben, zeker voor bedrijven is dit belangrijk. De missie van het beveiligingsmodel van Vault is:

- Vertrouwen
- Integriteit
- Beschikbaarheid
- Verantwoordelijkheid
- Authenticatie

In het kort betekent dit dat opgeslagen data en data die onderweg zijn veilig moeten zijn voor cybercriminelen. Gebruikers moeten ook op een juiste manier geauthenticeerd zijn om de data te kunnen opvragen of aanpassen. Alle interacties moeten worden opgeslagen of gelogd, zodat bij problemen de bron van het verzoek kan worden gevonden. Het systeem moet ook opgewassen zijn tegen pogingen om authenticatie te omzeilen. [22]

3.2.2.1 Bedreigingsmodel

Het bedreigingsmodel van Vault heeft verschillende oogmerken:

- Afluisteren van alle Vault-communicatie moet onmogelijk zijn. Gebruikerscommunicatie met Vault moet veilig zijn, communicatie tussen Vault en zijn opslag ook.
- Aanpassingen van data in rust of onderweg moeten detecteerbaar zijn en ervoor zorgen dat Vault stopt met het verwerken van deze data.
- Toegang moet alleen mogelijk zijn met de juiste authenticatie en autorisatie.
- Als het logboek is ingeschakeld, dienen alle aanvragen en antwoorden gelogd te worden voor de gebruiker de geheime data ontvangt.
- Toegang tot het geheime materiaal in het geval van een uitval moet mogelijk zijn. In de praktijk heeft Vault dus clustering ingebouwd.

De volgende aspecten maken geen deel uit van het model:

- Beveiliging tegen ongewenste acties op de opslag. Als een aanvaller hier toegang toe krijgt, kan deze data verwijderen waardoor Vault dataverlies kan lijden.
- De situatie waarin een crimineel toegang krijgt tot de opslag en er materiaal gelekt is. Deze kan dan nakijken of dit materiaal bestaat in de opslag.
- Als een aanvaller het geheugen van een machine waarop de Vault-server staat kan bekijken, kunnen de data worden aangetast. [22]

3.2.2.2 Externe beveiliging

Er zijn drie verschillende systemen die kwetsbaar kunnen zijn in een beveiligingscontext. De eerste is de applicatie voor de eindgebruiker die met Vault communiceert via een API, de tweede de Vault-server die de API aanbiedt, de laatste de opslag die de server gebruikt om te lezen en te schrijven.

Er is geen vertrouwen tussen de applicatie voor eindgebruikers en de Vault-server. De applicatie gebruikt TLS om de identiteit bij de server te verifiëren. De server vereist dat de applicatie zichzelf bij ieder verzoek authenticeert. Een applicatie die dit niet doet heeft geen toestemming om data op te vragen of in te loggen.

De opslag wordt ook niet vertrouwd door de Vault-server. Voordat data de Vault-server verlaten worden deze geëncrypteerd met een 256-bit *Advanced Encrypted Standard* (AES) sleutel. [22]

3.2.2.3 Interne beveiliging

Ook in het Vault-systeem is veiligheid belangrijk. Het moet voor een aanvaller onmogelijk zijn om toegang te krijgen tot materiaal waar hij bij wil maar waar hij niet bij mag. Zoals eerder gesteld moet een eindgebruiker zichzelf altijd verifiëren. Dit gebeurt bij iedere aanvraag. Vault kijkt dan na of dit verzoek geldig is, dus of het niet ingetrokken of op een andere manier ongeldig is. Daarna worden de juiste *access-control list* (ACL)-regels gegenereerd. Als er geen toegang is gegeven in deze regels, wordt de toegang geweigerd. Als er meerdere regels zijn voor een onderdeel, worden die met de hoogste rechten genomen. [22]

Als laatste heeft Vault ondersteuning voor '*Shamir's Secret Sharing technique*'. Vault start altijd '*sealed*'. Dit betekent dat de encryptiesleutel die nodig is om de opslag te openen nog niet bekend is. Om de Vault te openen moet de hoofdsleutel ingegeven worden. Door middel van '*Shamir's Secret Sharing technique*' kan deze opgesplitst worden in verschillende delen. Er zijn in Vault standaard drie van de vijf porties van de hoofdsleutel nodig om de Vault te openen. Het aantal porties dat nodig is of waarin wordt voorzien kan eventueel worden veranderd. [22] [23] [24]

3.2.3 Gebruik

Zoals in de inleiding al is aangegeven is het vaak moeilijk om veiligheid en gebruikersgemak te combineren. Het is natuurlijk zeer gewenst dat het ook makkelijk is om de software te gebruiken. Vault heeft zoals de meeste programma's gericht op eindgebruikers een UI. Deze bevindt zich volledig in de webbrowser. [25]

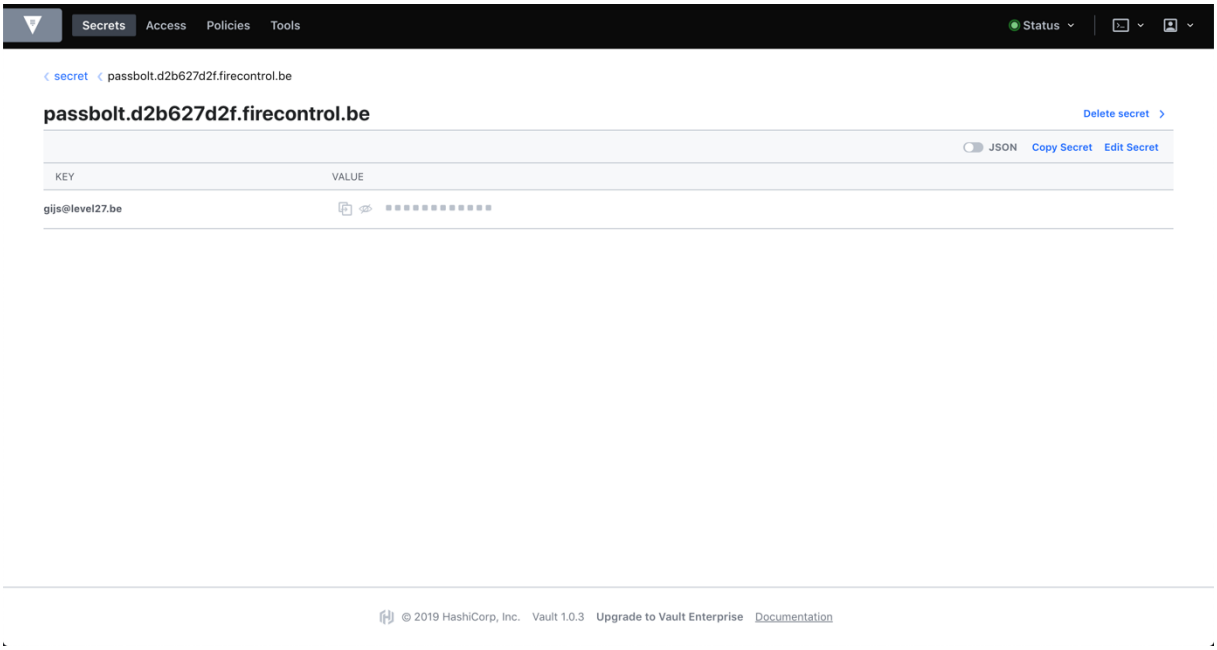
De UI is makkelijk in gebruik voor taken als het opvragen van gegevens. Dit kan door op al geconfigureerde '*secret engines*' te klikken en dan op de gewenste geheime gegevens. Als Vault moet worden geconfigureerd, kunnen de panelen *Access* en *Policies* gebruikt worden. Een nieuwe *secret engine* maken kan via de knop '*Enable new engine*'. Via de panelen *Access* en *Policies* kunnen veel opties geconfigureerd worden. Het is echter niet mogelijk om een nieuwe gebruiker aan te maken zonder een CLI-commando. Om dit toch zoveel mogelijk via de UI te laten verlopen volstaat een commando. De rest van de gebruikersconfiguratie kan via de UI.

Een goed overzicht van veelgebruikte functies en of die mogelijk zijn in de UI is te vinden in tabel 3.

Tabel 3 - Functionaliteit Vault UI

	UI	CLI
Gegevens opvragen	X	X
Gebruikers aanmaken		X
Toegang instellen voor gebruikers	X	X
Toegangsniveaus aanmaken (*policies*)		X
Secret engine aanmaken	X	X
Secret engine configureren/instellen		X
Nieuwe *secret* aanmaken	X	X

Zoals te zien in de bovenstaande tabel kan alles met de CLI worden opgelost, maar is het niet mogelijk om alles met de UI te doen. De UI is dus meer gericht op niet-technisch personeel dat bijvoorbeeld gegevens op wil vragen. Dit personeel kan ook kleine aanpassingen doen via de UI. Grote configuraties zullen echter nog steeds via de CLI moeten plaatsvinden. In figuur 13 wordt weergeven hoe gegevens kunnen worden opgevraagd uit Vault.



Figuur 13 - Gegevens uit Vault opvragen

3.2.4 Implementatie

Van Vault zijn verschillende onderdelen nodig om tot een succesvolle installatie te komen: de opslag en de Vault-server zelf. Voor de opslag prefereert Vault Consul, dit omdat Consul en Vault beide zijn gemaakt door Hashicorp. Er is een enorme hoeveelheid documentatie beschikbaar op de website van Vault. Ook is informatie te vinden over hoe Vault het beste kan worden geïnstalleerd. Verder is op Google veel informatie beschikbaar over dit onderwerp. Indien er problemen zijn, kan er dus altijd worden teruggevallen op documentatie. De installatie verloopt best vloeiend. Om een nieuwe '*secret engine*' op te zetten zal echter altijd wat onderzoek moeten worden gedaan naar hoe dit precies moet.

3.2.5 Nadelen

Vault heeft veel functionaliteiten en functies, maar het is vaak moeilijk om een volledig beeld te krijgen van hoe alles precies werkt. De UI heeft ook een CLI nodig om alle functies te kunnen gebruiken. Dit maakt het voor minder technische gebruikers heel moeilijk om de volledige functionaliteiten eenvoudig te gebruiken. Iedere '*secret engine*' vereist voor de installatie ook een andere *workflow*. Er is dus geen makkelijke manier om een nieuwe *engine* direct werkend te krijgen. Replicatie of *redundancy* van de applicatie vereist ook een '*Enterprise*' abonnement. Dit kost dus geld.

3.3 Psono

3.3.1 Functies en uitbreidingen

Psono houdt net als Vault wachtwoorden bij. In tegenstelling tot Vault worden enkel wachtwoorden bijgehouden, er is geen mogelijkheid om bijvoorbeeld gegevens van SSH bij te houden. Wel heeft Psono een browserextensie. Deze extensie vult automatisch wachtwoorden in op websites. Psono heeft ook ondersteuning voor een '*highly available cluster*'. Verder is het mogelijk om verschillende vormen van *caching* te gebruiken. *Caching* is het tijdelijk bijhouden van gegevens om deze sneller beschikbaar te stellen. Psono ondersteunt Redis, memcache en het lokaal geheugen. [26] Het is ook mogelijk om het *Lightweight Directory Access Protocol* (LDAP) te gebruiken om in te loggen op Psono. Zo kan er makkelijk gemigreerd worden naar de manager. Deze uitbreiding zit in het Enterprise-pakket en is dus betaald. [27]

3.3.2 Beveiliging

Psono zet eveneens in op beveiliging. De bestaansgrond van Psono is dat de ontwikkelaar graag een '*self hosted password manager*' had die ook veilig is. De waarden van Psono zijn:

- Goede beveiliging
- De mogelijkheid om het zelf te hosten
- Zo weinig mogelijk *bugs* [28]

3.3.2.1 Encryptie

Psono gebruikt Curve25519 en Salsa20 in de vorm van de *Networking and Cryptography library* (NaCl). Curve25519 is een vorm van Elliptische Curve Cryptografie (ECC) en is ontwikkeld om samen te gebruiken met Elliptische Curve Diffie-Hellman (ECDH). [29] De server gebruikt PyNaCl en de UI ecma-nacl. Beide zijn welgekende implementaties van NaCl. [30] De reden dat er niet gekozen werd voor Rivest Shamir Adleman (RSA) en AES is dat het makkelijk is om deze fout te implementeren en zo beveiligingslekken te creëren. [31] [32]

3.3.2.2 Browser encryptie

Psono versleutelt alle data met NaCl geheime-sleutel-authenticatie voordat de data de browser verlaten. De data worden daarna op de server gezet in de vorm van een "*secret object*". De sleutel voor het decrypteren en de encryptie wordt willekeurig gegenereerd. Deze wordt samen met metadata van de geheime sleutel die aangemaakt wordt tijdens de installatie in de opslag geplaatst. De browserapplicatie encrypteert deze opslag met een afleiding van het wachtwoord van de gebruiker. Het algoritme dat voor dit alles wordt gebruikt is scrypt. [31] [33]

3.3.2.3 Transport encryptie

Drie encryptielagen beschermen de data van en naar de webbrowser. De basislaag is de encryptie die in het vorige deel werd beschreven. Daarbovenop heeft Psono een middelste laag. Deze laag maakt een soort van *Virtual Private Network* (VPN)-tunnel tussen de website of browserextensie en de server. Als laatste wordt de *Secure Sockets Layer* (SSL) gebruikt. Psono vertelt dat *"both our mid and outer layer are built and configured to support perfect forward security [sic] (PFS)"* [31]

3.3.2.4 Encryptie bij opslag

De server encrypteert alle private data. Hieronder vallen bijvoorbeeld ook e-mailadressen van gebruikers. De opslag gebruikt de 'Salsa20 *stream cipher*' samen met een 'Poly1305 *Message Authentication Code* (MAC)'. [31]

3.3.2.5 Vulnerability scans

De ontwikkelaars van Psono hebben meerdere scanners geplaatst die lekken proberen te vinden in hun applicatie. Zo kunnen eventuele lekken snel worden opgelost. Deze scans worden ook automatisch toegepast op de *development pipeline*. Als er een *release* wordt gedetecteerd die kwetsbaar is, wordt deze geblokkeerd en niet gereleaset. Er worden ook derde partijen ingehuurd voor beveiligingschecks. [31]

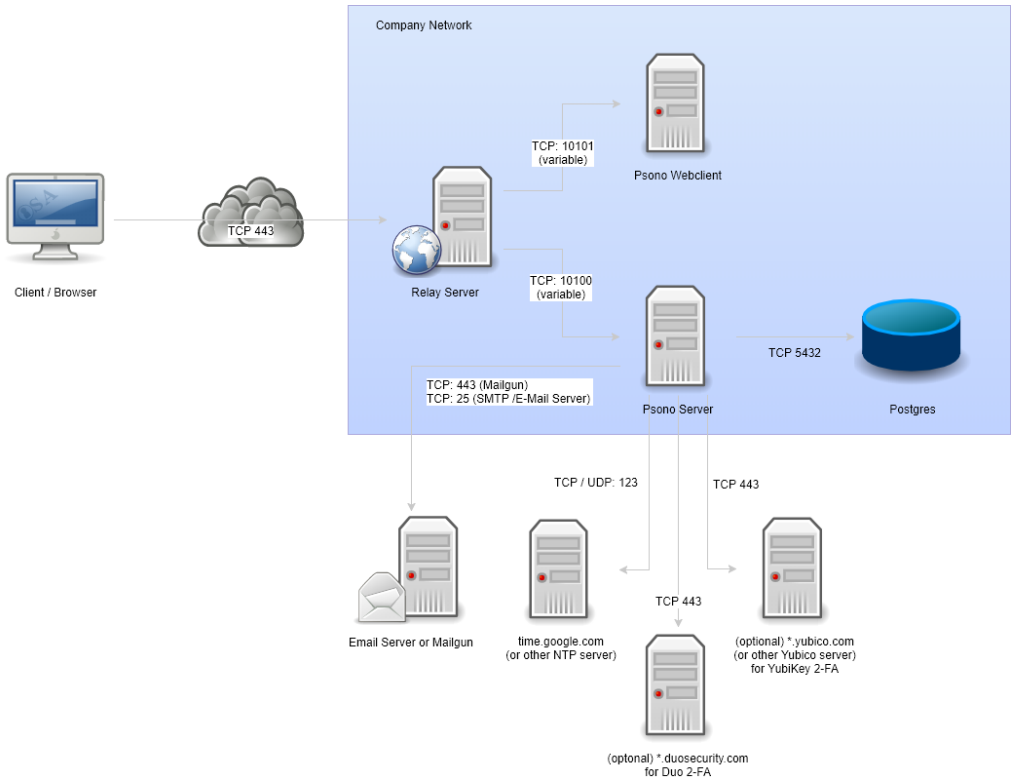
3.3.3 Gebruik

Psono bestaat uit drie onderdelen: de gebruikersinterface, de administratie-interface en de server. Gebruikers worden aangemaakt via de CLI. Het is ook mogelijk om dit via de gebruikersinterface te doen, maar dit vereist dat een e-mailserver is geconfigureerd. Als er is ingelogd, kunnen wachtwoorden of tekst in de *Datastore* worden opgeslagen. Deze kunnen ook gedeeld worden met andere gebruikers door een directe *share* als de e-mail van de een andere gebruiker bekend is of als deze uit een groep bestaat.

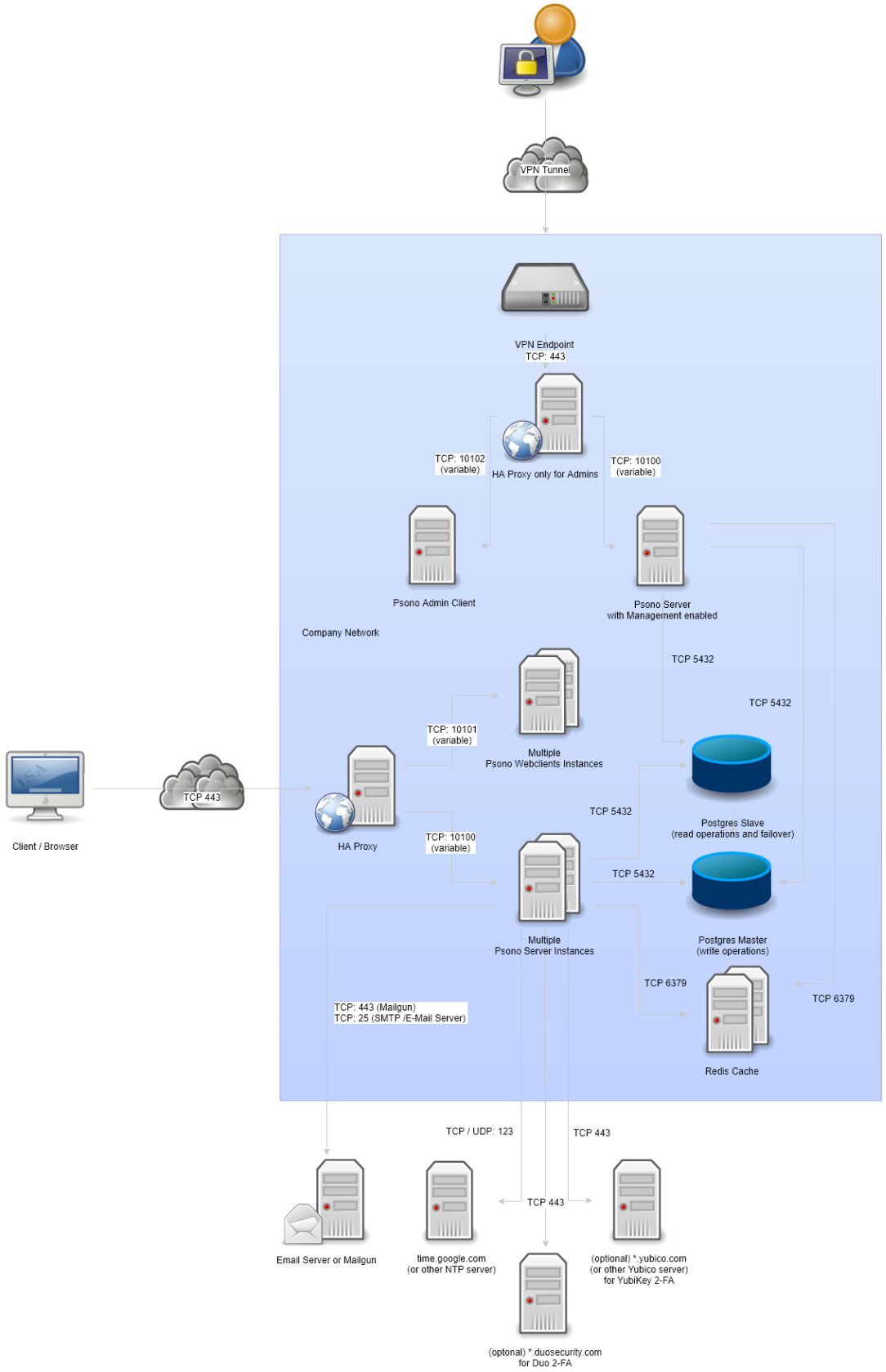
Er is ook een optionele extensie beschikbaar voor webbrowsers. Als er via deze extensie wordt ingelogd, wordt iedere keer bij de ingave van geheime gegevens een dialoog weergeven. Door gebruik van deze dialoog kunnen deze gegevens veilig worden opgeslagen in Psono. De extensie biedt dan de optie om deze gegevens in te vullen in hetzelfde inlogvenster.

3.3.4 Implementatie

Psono heeft een *postgress* database nodig. Een andere database kan niet gebruikt worden omdat Psono specifieke extensies van *postgress* nodig heeft. De server kan *´bare bones´* geïnstalleerd worden, maar het wordt aangeraden om dit via Docker te doen, omdat de verschillende onderdelen dan makkelijker samenwerken. Psono bestaat uit drie onderdelen, de Psono-server, *webclient* en admin *websclient*. Om deze op een goede manier te laten samenwerken is ook een nginx *reverse proxy* gewenst. De documentatie voor de installatie doorloopt iedere stap van het proces. Op figuur 14 en 15 worden verschillende mogelijke configuraties voor Psono weergeven. Deze zijn te vinden op de volgende twee pagina's.



Figuur 14 - Psono basis-set-up [26]



Figuur 15 - Geavanceerde Psono set-up [26]

3.3.5 Nadelen

Om Psono te clusteren dient een licentie gekocht te worden. Verder neemt de installatie meer stappen in beslag dan de andere twee *password managers* en is deze complexer. Dit werd toegelicht in het vorige deel: implementatie. Psono heeft ook niet zoveel functies als Vault. Het houdt alleen maar wachtwoorden en tekst veilig bij.

3.4 Passbolt

3.4.1 Functies en uitbreidingen

Als laatste is er Passbolt. Deze manager is nog niet uitontwikkeld, waardoor constant nieuwe *features* worden toegevoegd. De belangrijkste *features* tot nu toe zijn onder andere een administratiepaneel, LDAP-integratie, het importeren van wachtwoorden uit andere managers en het labelen van wachtwoorden.

Belangrijke features die nog moeten uitkomen zijn het automatisch aanvullen van wachtwoorden, een log bijhouden, gebruikersrechten aanpassen, mappen aanmaken met gegevens, Passbolt offline gebruiken en bestanden bijhouden. [34]

3.4.2 Beveiliging

Passbolt bevindt zich nog steeds in een 'bèta'-versie. Dit betekent dat de huidige versie in constante ontwikkeling is en nog incompleet is. Er is echter al veel moeite in de beveiliging gestoken. In dit deel wordt uitgelegd welke beveiligingsfunctionaliteit Passbolt al heeft. [35]

3.4.2.1 Encryptie

Er zijn drie soorten data die Passbolt moet beschermen. Dit zijn:

- De data bij transport
- De data in het geheugen
- De data niet in gebruik op het bestandsysteem

Op dit moment wordt bij de data in het geheugen alleen het wachtwoord geëncrypteerd. De gebruikersnaam, reacties of lijst met mensen met wie een wachtwoord gedeeld wordt zijn niet geëncrypteerd met OpenPGP. De wachtwoorden zelf zullen echter altijd op een gegeven moment beschikbaar zijn in niet-geëncrypteerde vorm. Anders zouden deze niet gebruikt kunnen worden.

Voor data tijdens transport zijn alle data geëncrypteerd met SSL, voor het overige zijn er geen maatregelen genomen. Er moet dus zeker in een SSL-certificaat voorzien zijn.

Voor data op het bestandsysteem is de database met alle gegevens geëncrypteerd. Passbolt laat ook weten dat "*it is also possible to encrypt the database at the file system level as well. This will add another encryption layer that can be useful, for example, in the case where the machine running passbolt is seized or stolen*." [36]

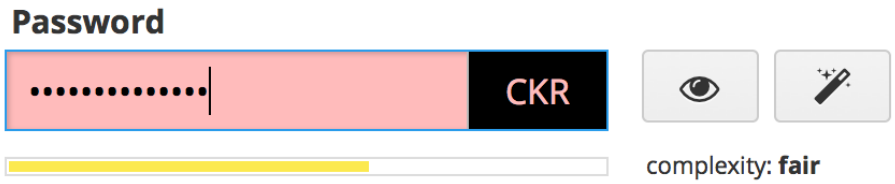
3.4.2.2 Soort encryptie

Passbolt-servers slaan wachtwoorden nooit op in platte tekst. Alle wachtwoorden zijn geëncrypteerd bij de cliënt met behulp van de browserextensie. De browserextensie gebruikt openPGP. OpenPGP is een standaard die een sterk PKI-systeem aanbiedt samen met symmetrische cryptografie. De private sleutel die gebruikt wordt om het wachtwoord te decrypteren is zelf ook versleuteld met een wachtwoord. Op de server wordt de GnuPG-php-extensie gebruikt, de openpgp-php om de publieke

sleutel te valideren en om GPGAuth-authenticatie te ondersteunen. [37] GPGAuth wordt dus gebruikt om in te loggen in Passbolt. [38]

3.4.2.3 Security token

Een security *token* wordt gebruikt om *phishing* tegen te gaan. *Phishing* is een vorm van internetfraude. Het bestaat uit het oplichten van mensen door ze te lokken naar een valse website die een kopie is van de echte website. [39] Op deze manier kunnen ze in het geval van Passbolt wachtwoorden bemachtigen. Een *security token* is gevisualiseerd in figuur 16.

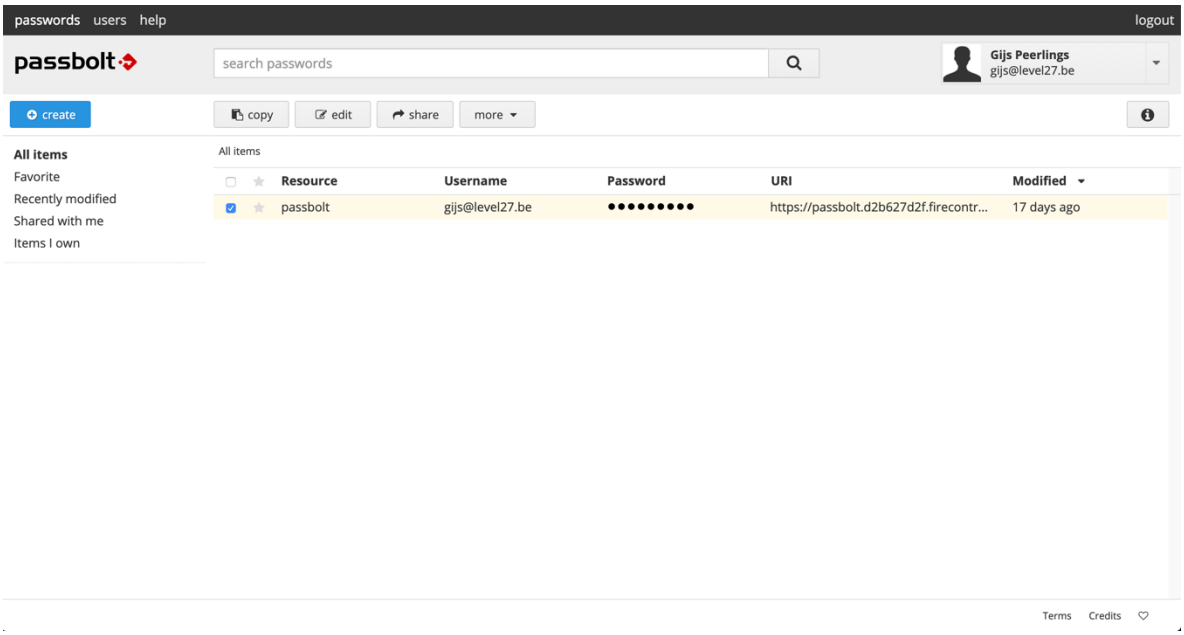


Figuur 16 - Passbolt security token [40]

Wanneer er sensitieve gegevens worden ingegeven, kan de gebruiker op deze manier zien dat de website niet nep is. De letters CKR en de rode kleur zijn voor deze specifieke gebruiker altijd hetzelfde. [40]

3.4.3 Gebruik

Om Passbolt te gebruiken is de browserextensie van Passbolt vereist. Dit is nodig om meer cryptografische functionaliteit aan te bieden. Meer hierover kan gevonden worden in het deel over beveiliging (4.4.2). Bij het eerste gebruik wordt gevraagd een *security token* te maken. Deze blijft altijd hetzelfde. Nadat er is ingelogd kunnen er wachtwoorden worden toegevoegd. Op dit moment kunnen deze alleen maar worden opgeslagen. De extensie probeert deze ook in te vullen in tekstvelden, maar dit lukt nog niet altijd zo goed. De wachtwoorden kunnen ook gedeeld worden met andere gebruikers. In de administratie-interface kunnen ook makkelijk nieuwe gebruikers worden toegevoegd. Onder het profiel van de gebruiker kan de publieke en private sleutel worden gedownload. Ook kunnen er rechten worden ingesteld op de wachtwoorden. Figuur 17 geeft weer hoe Passbolt eruitziet.



Figuur 17 - Interface Passbolt

3.4.4 Implementatie

Passbolt heeft net als Psono een database nodig. In het geval van Passbolt kan er gekozen worden tussen eender welke database. Indien er al een database is, kan Passbolt geïnstalleerd worden met een simpel Docker-commando. Aangezien er nog geen gebruiker is, moet deze via de *command line* worden aangemaakt. De rest kan allemaal via de UI. De installatie verloopt dus zonder al te veel moeite en complicaties. [41]

3.4.5 Nadelen

Passbolt heeft aanzienlijk minder functies dan de andere twee wachtwoordbeheerders. Het enige wat op dit moment mogelijk is, is wachtwoorden bijhouden. Er zijn dus nog geen extra functionaliteiten. Dit zal gedurende het komende jaar wel veranderen. De beveiliging is ook op een minder sterke manier opgebouwd dan die van de andere producten. Deze zal ook worden doorontwikkeld. [42]

3.5 Conclusie

In dit deel wordt het volledige onderzoek samengebracht. De drie wachtwoordbeheerders worden met elkaar vergeleken. Er wordt een antwoord gezocht op de deelvragen gedefinieerd in het deel over de onderzoeksvraag. De vragen zijn de volgende:

- Welke veiligheidsmaatregelingen zijn ingebouwd en hoe werken deze precies?
- Welke wachtwoordbeheerder is het makkelijkst in gebruik?
- Wie heeft de meeste functionaliteit?
- Wat is het makkelijkst op te zetten?

3.5.1 Beveiliging

Zoals beschreven in het vorige gedeelte heeft iedere *password manager* veiligheidsmaatregelingen ingebouwd. Vault gebruikt een systeem waarbij ACL-rechten worden gebruikt om toegang te krijgen tot gegevens. Door middel van een log-in wordt gekeken welke gebruiker welke rechten heeft. De data worden ook beveiligd met '*Shamir's Secret Sharing technique*'.

Psono gebruikt een hele reeks aan encryptietechnieken om de data veilig te houden. Deze worden voornamelijk voorzien door *open-source 'libraries*'. Ook doet Psono '*vulnerability scans*' om te kijken of een nieuwe '*release*' van de software wel veilig is.

Als laatste is er Passbolt. Deze vereist een browserextensie om meer cryptografische mogelijkheden te bieden. Er wordt voornamelijk gebruikgemaakt van *Pretty Good Privacy* (PGP). Ook heeft Passbolt als enige een *security token* om *phishing* tegen te gaan.

3.5.2 Gebruik

Vault heeft net als de andere *password managers* een UI, maar bij Vault kan deze niet alle functionaliteit gebruiken. Er is vaak nog een terminal nodig. Voor een minder technische gebruiker kan dit best gebruiksonvriendelijk aanvoelen.

Psono omvat een volledige UI, die alle mogelijkheden kan gebruiken. De UI is opgesplitst in twee delen, een administratie-*interface* en een gebruikers*interface*. In de administratie-*interface* kunnen onder andere statistieken worden opgevraagd. Psono vult ook goed wachtwoordvelden in als de extensie is geïnstalleerd. Dit is niet vereist.

In Passbolt is het gebruik van een extensie wel vereist. Dit voorziet betere encryptiemogelijkheden, maar verlaagt natuurlijk het gebruiksgemak. De extensie probeert ook wachtwoordvelden in te vullen. Dit werkt echter nog niet altijd. Bij twintig pogingen slaagde Passbolt er maar zes keer in om het wachtwoord correct en op de juiste plaats in te vullen.

3.5.3 Functies en uitbreidingen

Vault heeft veel extra functies buiten het bijhouden van wachtwoorden. Deze komen in de vorm van '*secret engines*'. Zo zijn er SSH-gegevens, Google *Cloud*, Amazon AWS en nog veel meer. Deze zijn allemaal makkelijk in gebruik en goed geïntegreerd. Vault biedt ook ondersteuning om makkelijk een *highly available* server te installeren. Psono en Passbolt hebben beide alleen ondersteuning voor wachtwoorden, bij deze twee is er geen makkelijke manier om een cluster op te zetten.

3.5.4 Implementatie

Vault heeft twee onderdelen nodig om te kunnen functioneren: een opslag en Vault zelf. Ook is er voor de opzet van een nieuwe *secret engine* altijd research en implementatie nodig. Sommige functies die makkelijk te gebruiken zijn in de andere twee *password managers* zijn in Vault moeilijk in gebruik.

Psono heeft drie delen nodig om te werken: een server voor Psono, een webserver voor de gebruikers*interface* en een webserver voor de administrators. Hiervoor wordt vaak Docker gebruikt, zodat er geen drie verschillende machines nodig zijn. Ook is een complexe nginx-configuratie vereist. Hier is wel een *template* voor te vinden, maar die moet nog op verschillende manieren worden aangepast.

Als laatste is er Passbolt. Deze is geïnstalleerd met drie commando's. Een nginx *reverse proxy* is optioneel. De configuratie hiervoor is terug te vinden in bijlage B. Deze omvat ook de configuratie voor Psono.

3.5.5 Vergelijking

Nu alle *password managers* zijn onderzocht kan er een vergelijkingsmatrix worden opgesteld op basis van de deelvragen. Deze is te vinden in tabel 4. Hier wordt weergeven welke *password manager* elke deelvraag het beste oplost.

Tabel 4 - Vergelijkingsmatrix

	Vault	Psono	Passbolt
Beveiliging		X	
Gebruik		X	
Functionaliteit	X		
Implementatie			X

Zoals te zien in de bovenstaande tabel heeft Psono twee van de vier deelvragen het beste opgelost. Dit wil natuurlijk niet zeggen dat dit ook direct de beste *password manager* is. De keuze voor een van de drie *password managers* hangt af van de noden van het bedrijf. Level27 heeft gekozen voor Vault omdat deze de meeste functionaliteit heeft.

Bibliografie

[1] Docker Inc., „What is a Container?," [Online]. Available: https://www.docker.com/resources/what-container. [Geopend 28 02 2019].

[2] Docker Inc., „Overview of Docker Compose," [Online]. Available: https://docs.docker.com/compose/overview/. [Geopend 28 02 2019].

[3] M. Rouse, „What is Docker image?," TechTarget, 06 2018. [Online]. Available: https://searchitoperations.techtarget.com/definition/Docker-image. [Geopend 28 02 2019].

[4] Hashicorp, "Hashicorp homepage," Hashicorp, 2019. [Online]. Available: https://www.hashicorp.com/.

[5] W. Morgan, „What's a service mesh? And why do I need one?," 25 04 2017. [Online]. Available: https://blog.buoyant.io/2017/04/25/whats-a-service-mesh-and-why-do-i-need-one/. [Geopend 01 03 2019].

[6] HashiCorp, „Introduction to Consul," [Online]. Available: https://www.consul.io/intro/index.html. [Geopend 01 03 2019].

[7] B. Ward, „Install Vault on Ubuntu 18.04," 13 11 2018. [Online]. Available: https://www.admintome.com/blog/install-vault-on-ubuntu-18-04/. [Geopend 01 03 2019].

[8] HashiCorp, „Identity: Entities and Groups," [Online]. Available: https://learn.hashicorp.com/vault/identity-access-management/iam-identity. [Geopend 04 03 2019].

[9] Hashicorp, „Policies," Hashicorp, [Online]. Available: https://www.vaultproject.io/docs/concepts/policies.html. [Geopend 02 04 2019].

[10] K. J. Hazel Virdó, „How To Secure Nginx with Let's Encrypt on Ubuntu 18.04," DigitalOcean Inc., 27 04 2018. [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-18-04. [Geopend 04 03 2019].

[11] Psono, „Install Postgres for Psono," 03 02 2019. [Online]. Available: https://doc.psono.com/mydoc_install_postgres.html. [Geopend 05 03 2019].

[12] Psono, „Install Psono Server CE," 03 02 2019. [Online]. Available: https://doc.psono.com/mydoc_install_server.html. [Geopend 05 03 2019].

[13] Psono, „Install Psono Webclient," 03 02 2019. [Online]. Available: https://doc.psono.com/mydoc_install_webclient.html. [Geopend 05 03 2019].

[14] Psono, „Install Psono Admin Webclient," 03 02 2019. [Online]. Available: https://doc.psono.com/mydoc_install_admin_webclient.html. [Geopend 05 03 2019].

[15] Hashicorp, „Consensus Protocol,” [Online]. Available: https://www.consul.io/docs/internals/consensus.html#deployment-table. [Geopend 04 04 2019].

[16] A. Umerov, „Generating SSH One-Time Passwords with Vault,” 27 10 2018. [Online]. Available: https://medium.com/@Amet13/generating-ssh-one-time-passwords-with-vault-7aa7fab0032.

[17] ManageEngine, „What is password management?,” Zoho Corp., 2019. [Online]. Available: https://www.manageengine.com/products/passwordmanagerpro/what-is-password-management.html. [Geopend 28 02 2019].

[18] V. Ng, „Biometric Authentication in iOS,” Medium, [Online]. Available: https://medium.com/@vishwasng/bibiometric-authentication-in-ios-36f51a65b39b. [Geopend 28 02 2019].

[19] J. McKinnon, „Why Two-Factor Authentication Isn’t Always Totally Secure,” Incsub, 27 03 2018. [Online]. Available: https://premium.wpmudev.org/blog/why-two-factor-authentication-isnt-always-totally-secure/. [Geopend 04 03 2019].

[20] HashiCorp homepage, [Online]. Available: https://www.vaultproject.io/docs/secrets/index.html. [Geopend 05 03 2019].

[21] HashiCorp, „High Availability Mode,” [Online]. Available: https://www.vaultproject.io/docs/concepts/ha.html. [Geopend 05 03 2019].

[22] HashiCorp, „Security Model,” [Online]. Available: https://www.vaultproject.io/docs/internals/security.html. [Geopend 06 03 2019].

[23] Wikimedia Foundation, Inc., „Shamir's Secret Sharing,” [Online]. Available: https://en.wikipedia.org/wiki/Shamir's_Secret_Sharing. [Geopend 06 03 2019].

[24] P. Feldman, „A practical scheme for non-interactive verifiable secret sharing,” IEEE, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4568297. [Geopend 06 03 2019].

[25] HashiCorp, „Web UI,” [Online]. Available: https://learn.hashicorp.com/vault/getting-started/ui. [Geopend 06 03 2019].

[26] S. Pfeiffer, „Admin Documentation of Psono,” [Online]. Available: https://doc.psono.com/mydoc_index.html. [Geopend 06 03 2019].

[27] S. Pfeiffer, „Features for Admins,” [Online]. Available: https://doc.psono.com/mydoc_overview_supported_features.html. [Geopend 07 03 2019].

[28] S. Pfeiffer, „About Psono,” [Online]. Available: https://doc.psono.com/mydoc_overview_about.html. [Geopend 07 03 2019].

[29] Wikimedia Foundation, Inc., „Curve25519,” [Online]. Available: https://en.wikipedia.org/wiki/Curve25519. [Geopend 07 03 2019].

[30] Y. Acar, M. Backes, S. Fahl, S. Garfinkel, D. Kim, M. L. Mazurek and C. Stransky, "Comparing the Usability of Cryptographic APIs," IEEE, San Jose, CA, USA, 2017.

[31] S. Pfeiffer, „Security of Psono," [Online]. Available: https://psono.com/security. [Geopend 07 03 2019].

[32] L. Pustina, „Crypto is Broken or How to Apply Secure Crypto as a Developer," Codecentric, 03 05 2014. [Online]. Available: https://blog.codecentric.de/en/2014/03/crypto-broken-apply-secure-crypto-developer/. [Geopend 07 03 2019].

[33] C. Percival en S. Josefsson, „RFC 7914," 08 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7914. [Geopend 07 03 2019].

[34] Passbolt, „Passbolt roadmap," [Online]. Available: https://www.passbolt.com/roadmap. [Geopend 08 04 2019].

[35] Passbolt, „Is it secure to use passbolt in its current version?," [Online]. Available: https://help.passbolt.com/faq/security/is-passbolt-secure. [Geopend 11 03 2019].

[36] Passbolt, „What data is encrypted in passbolt?," [Online]. Available: https://help.passbolt.com/faq/security/what-is-encrypted. [Geopend 11 03 2019].

[37] Passbolt, „What kind of encryption does passbolt use?," [Online]. Available: https://help.passbolt.com/faq/security/encryption-tech. [Geopend 11 03 2019].

[38] Passbolt, „How does authentication work in passbolt?," [Online]. Available: https://help.passbolt.com/faq/security/authentication. [Geopend 11 03 2019].

[39] Wikimedia Foundation, Inc., „Phishing," [Online]. Available: https://nl.wikipedia.org/wiki/Phishing. [Geopend 11 03 2019].

[40] Passbolt, „What is the security token?," [Online]. Available: https://help.passbolt.com/faq/security/security-token. [Geopend 11 03 2019].

[41] Passbolt, „Docker passbolt installation," [Online]. Available: https://help.passbolt.com/hosting/backup. [Geopend 11 03 2019].

[42] Passbolt, „Is it secure to use passbolt in its current version?," [Online]. Available: https://help.passbolt.com/faq/security/is-passbolt-secure. [Geopend 11 03 2019].

[43] A. Grover, „Basics Of Docker," 13 10 2017. [Online]. Available: https://codeburst.io/basics-of-docker-c1416b02d03c. [Geopend 04 03 2019].

[44] R. Natário, „High Availability - Terminology (II)," 02 2011. [Online]. Available: https://networksandservers.blogspot.com/2011/02/high-availability-terminology-ii.html. [Geopend 04 03 2019].

Bijlagen

A. Admin ACL policy

Hieronder staat een *policy* die te gebruiken is in Vault om een *Administrator* gebruiker te maken. Deze gebruiker kan de belangrijkste instellingen aanpassen, maar heeft minder rechten dan de *root token*. Deze bijlage wordt gebruikt in hoofdstuk 3.4.1.6.

```
# Manage auth methods broadly across Vault

path "auth/*"
{
  capabilities = ["create", "read", "update", "delete", "list", "sudo"]
}

# Create, update, and delete auth methods
path "sys/auth/*"
{
  capabilities = ["create", "update", "delete", "sudo"]
}

# List auth methods
path "sys/auth"
{
  capabilities = ["read"]
}

# List existing policies
path "sys/policy"
{
  capabilities = ["read"]
}

# Create and manage ACL policies via CLI
path "sys/policy/*"
{
  capabilities = ["create", "read", "update", "delete", "list", "sudo"]
}

# Create and manage ACL policies via API & UI
path "sys/policies/acl/*"
{
  capabilities = ["create", "read", "update", "delete", "list", "sudo"]
}

# List, create, update, and delete key/value secrets
path "secret/*"
{
  capabilities = ["create", "read", "update", "delete", "list", "sudo"]
}

# Manage secret engines
path "sys/mounts/*"
{
  capabilities = ["create", "read", "update", "delete", "list", "sudo"]
}
```

```
# List existing secret engines.
path "sys/mounts"
{
  capabilities = ["read"]
}

# Read health checks
path "sys/health"
{
  capabilities = ["read", "sudo"]
}

path "ssh/creds/otp_key_role" {
  capabilities = [ "update" ]
}
```

B. Nginx configuratie

De nginx configuratie om Passbolt en Psono uit te voeren. Deze configuratie is geoptimaliseerd voor letsencrypt. Indien een van de twee *password managers* niet gebruikt wordt, kan deze configuratie makkelijk bijgewerkt worden door het deel van deze *manager* te verwijderen.

```
server {
   listen 80;
   server_name d2b627d2f.firecontrol.be;
   return 301 https://$host$request_uri;
}

server {
   listen 80;
   server_name www.d2b627d2f.firecontrol.be;
   return 301 https://$host$request_uri;
}

server {
   if ($host = passbolt.d2b627d2f.firecontrol.be) {
      return 301 https://$host$request_uri;
   } # managed by Certbot


   listen 80;
   server_name passbolt.d2b627d2f.firecontrol.be;
   return 301 https://$host$request_uri;
         location ^~ /.well-known/acme-challenge/ {
         default_type "text/plain";
         root /var/www/html;
       }
}

server {
   listen 443 ssl http2;
   server_name d2b627d2f.firecontrol.be;
   return 301 https://www.$host$request_uri;

   ssl_protocols TLSv1.2;
   ssl_prefer_server_ciphers on;
   ssl_session_cache shared:SSL:10m;
   ssl_session_tickets off;
   ssl_stapling on;
   ssl_stapling_verify on;
   ssl_session_timeout 1d;
   resolver 8.8.8.8 8.8.4.4 valid=300s;
   resolver_timeout 5s;
   ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-
CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-
RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-
AES128-SHA256:ECDHE-RSA-AES128-SHA256';

   add_header Strict-Transport-Security "max-age=63072000; includeSubdomains; preload";

   add_header Referrer-Policy same-origin;
   add_header X-Frame-Options DENY;
```

```
    add_header X-Content-Type-Options nosniff;
    add_header X-XSS-Protection "1; mode=block";

    # If you have the admin webclient installed too behind this reverse proxy domain, then you have to change the following from:
    # "connect-src 'self' https://api.pwnedpasswords.com;" to "connect-src 'self' https://example.com https://static.psono.com https://api.pwnedpasswords.com;"
    add_header Content-Security-Policy "default-src 'none';  manifest-src 'self'; connect-src 'self' https://d2b627d2f.firecontrol.be https://static.psono.com https://api.pwnedpasswords.com; font-src 'self'; img-src 'self' data:; script-src 'self'; style-src 'self' 'unsafe-inline'; object-src 'self'";

    ssl_certificate /etc/letsencrypt/live/d2b627d2f.firecontrol.be/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/d2b627d2f.firecontrol.be/privkey.pem;
}
server {
    listen 443 ssl http2;
    server_name www.d2b627d2f.firecontrol.be;

    ssl_protocols TLSv1.2;
    ssl_prefer_server_ciphers on;
    ssl_session_cache shared:SSL:10m;
    ssl_session_tickets off;
    ssl_stapling on;
    ssl_stapling_verify on;
    ssl_session_timeout 1d;
    resolver 8.8.8.8 8.8.4.4 valid=300s;
    resolver_timeout 5s;
    ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256';

    add_header Referrer-Policy same-origin;
    add_header X-Frame-Options DENY;
    add_header X-Content-Type-Options nosniff;
    add_header X-XSS-Protection "1; mode=block";

    # If you have the admin webclient installed too behind this reverse proxy domain, then you have to change the following from:
    # "connect-src 'self' https://api.pwnedpasswords.com;" to "connect-src 'self' https://example.com https://static.psono.com https://api.pwnedpasswords.com;"
    add_header Content-Security-Policy "default-src 'none';  manifest-src 'self'; connect-src 'self' https://d2b627d2f.firecontrol.be https://static.psono.com https://api.pwnedpasswords.com; font-src 'self'; img-src 'self' data:; script-src 'self'; style-src 'self' 'unsafe-inline'; object-src 'self'";

    ssl_certificate /etc/letsencrypt/live/d2b627d2f.firecontrol.be/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/d2b627d2f.firecontrol.be/privkey.pem;

    client_max_body_size 10m;

    gzip on;
    gzip_disable "msie6";

    gzip_vary on;
    gzip_proxied any;
    gzip_comp_level 6;
    gzip_buffers 16 8k;
```

```
    gzip_http_version 1.1;
    gzip_min_length 256;
    gzip_types text/plain text/css application/json application/x-javascript application/javascript text/xml
application/xml application/xml+rss text/javascript application/vnd.ms-fontobject application/x-font-ttf
font/opentype image/svg+xml image/x-icon;

    root /var/www/html;

    location /server {
        rewrite ^/server/(.*) /$1 break;
        proxy_set_header       Host $host;
        proxy_set_header       X-Real-IP $remote_addr;
        proxy_set_header       X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header       X-Forwarded-Proto $scheme;

        add_header Last-Modified $date_gmt;
        add_header Pragma "no-cache";
        add_header Cache-Control "private, max-age=0, no-cache, no-store";
        if_modified_since off;
        expires off;
        etag off;

        proxy_pass         http://localhost:10100;
    }

    location ~* ^/portal.*\.(?:ico|css|js|gif|jpe?g|png)$ {
        expires 30d;
        add_header Pragma public;
        add_header Cache-Control "public";

        # Comment in the following lines if you have the admin webclient running in a docker container
         proxy_set_header       Host $host;
         proxy_set_header       X-Real-IP $remote_addr;
         proxy_set_header       X-Forwarded-For $proxy_add_x_forwarded_for;
         proxy_set_header       X-Forwarded-Proto $scheme;
        #
         proxy_pass         http://localhost:10102;
         proxy_redirect      http://localhost:10102 https://d2b627d2f.firecontrol.be;
    }

    location ~* \.(?:ico|css|js|gif|jpe?g|png)$ {
        expires 30d;
        add_header Pragma public;
        add_header Cache-Control "public";

         #Comment in the following lines if you have the webclient running in a docker container
         proxy_set_header       Host $host;
         proxy_set_header       X-Real-IP $remote_addr;
         proxy_set_header       X-Forwarded-For $proxy_add_x_forwarded_for;
         proxy_set_header       X-Forwarded-Proto $scheme;

         proxy_pass         http://localhost:10101;
         proxy_redirect      http://localhost:10101 https://d2b627d2f.firecontrol.be;
    }

    # Comment in the following lines if you have the admin webclient running in a docker container
     location /portal {
```

```
        proxy_set_header       Host $host;
        proxy_set_header       X-Real-IP $remote_addr;
        proxy_set_header       X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header       X-Forwarded-Proto $scheme;
  #
        proxy_read_timeout  90;
  #
        proxy_pass         http://localhost:10102;
        proxy_redirect      http://localhost:10102 https://d2b627d2f.firecontrol.be;
   }

  # Comment in the following lines if you have the webclient running in a docker container
   location / {
        proxy_set_header       Host $host;
        proxy_set_header       X-Real-IP $remote_addr;
        proxy_set_header       X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header       X-Forwarded-Proto $scheme;

        proxy_pass         http://localhost:10101;
        proxy_read_timeout  90;

        proxy_redirect      http://localhost:10101 https://d2b627d2f.firecontrol.be;
   }
   }
server {
  listen 443 ssl;
  server_name passbolt.d2b627d2f.firecontrol.be;
#   ssl_certificate     /etc/letsencrypt/live/radarr.finestbit.com/fullchain.pem;
#   ssl_certificate_key /etc/letsencrypt/live/radarr.finestbit.com/privkey.pem;
       location / {
      proxy_pass        https://localhost:444;
      proxy_connect_timeout       300;
      proxy_send_timeout          300;
      proxy_read_timeout          300;
      send_timeout                300;
      proxy_redirect     off;
      proxy_set_header   Host $host;
      proxy_set_header   X-Real-IP $remote_addr;
      proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header   X-Forwarded-Host $server_name;
  }

  ssl_certificate /etc/letsencrypt/live/passbolt.d2b627d2f.firecontrol.be/fullchain.pem; # managed by
Certbot
  ssl_certificate_key /etc/letsencrypt/live/passbolt.d2b627d2f.firecontrol.be/privkey.pem; # managed
by Certbot
}
```

C. Psono configuratie

Hieronder staat het configuratiebestand voor Psono. De gegevens dienen aangevuld te worden aan de hand van de instructies in deel 3.4.3.3. Alle gegevens zijn al ten voorbeeld ingevuld, maar dienen natuurlijk aangepast te worden aan de huidige situatie.

```
# generate the following six parameters with the following command
# docker run --rm -ti psono/psono-server:latest python3 ./psono/manage.py generateserverkeys
SECRET_KEY: 'Dn%|6Qk%S]3"w("\\>P!SkwKoy!#;:3ZGTyzW%,py)w^&Hf{s8S'
ACTIVATION_LINK_SECRET: '&-JE2Rx6SOzUlAML>0P8}yGP%lx~6{RjqyG>FVjb1`hD9<&bhv'
DB_SECRET: 'J+KEk{%9wX`VqY\\j`\\JKnOU-IWh5ri9T03UPfdbl12VtFk.;,+'
EMAIL_SECRET_SALT: '$2b$12$o/0ibC18v1/.uDCJ5Q6BUu'
PRIVATE_KEY: '779f64c89ee8cf7649ad3653d4a8a33529013ba68b6327f02cc64484dddc14f0'
PUBLIC_KEY: '07a7b09216b3d3926fcd83e5e48d4e5448a30f3d8c703a8e25a6aca16420e854'

# The URL of the web client (path to e.g activate.html without the trailing slash)
WEB_CLIENT_URL: 'https://www.d2b627d2f.firecontrol.be'

# Switch DEBUG to false if you go into production
DEBUG: False

# Adjust this according to Django Documentation https://docs.djangoproject.com/en/1.10/ref/settings/
ALLOWED_HOSTS: ['*']

# Should be your domain without "www.". Will be the last part of the username
ALLOWED_DOMAINS: ['level27.be']

# If you want to disable registration, you can comment in the following line
# ALLOW_REGISTRATION: False

# If you want to restrict registration to some email addresses you can specify here a list of domains to
filter
# REGISTRATION_EMAIL_FILTER: ['company1.com', 'company2.com']

# Should be the URL of the host under which the host is reachable
# If you open the url you should have a text similar to {"detail":"Authentication credentials were not
provided."}
HOST_URL: 'https://www.d2b627d2f.firecontrol.be/server'

# The email used to send emails, e.g. for activation
EMAIL_FROM: ''
EMAIL_HOST: ''
EMAIL_HOST_USER: ''
EMAIL_HOST_PASSWORD : ''
EMAIL_PORT: 587
EMAIL_SUBJECT_PREFIX: ''
EMAIL_USE_TLS: True
EMAIL_USE_SSL: False
EMAIL_SSL_CERTFILE:
EMAIL_SSL_KEYFILE:
EMAIL_TIMEOUT:

# In case one wants to use mailgun, comment in below lines and provide the mailgun access key and
server name
# EMAIL_BACKEND: 'anymail.backends.mailgun.EmailBackend'
```

```
# MAILGUN_ACCESS_KEY: ''
# MAILGUN_SERVER_NAME: ''

# In case you want to offer Yubikey support, create a pair of credentials here
https://upgrade.yubico.com/getapikey/
# and update the following two lines before commenting them in
# YUBIKEY_CLIENT_ID: '123456'
# YUBIKEY_SECRET_KEY: '8I65IA6ASDFIUHGIH5021FKJA='

# If you have own Yubico servers, you can specify here the urls as a list
# YUBICO_API_URLS: ['https://api.yubico.com/wsapi/2.0/verify']

# Cache enabled without belows Redis may lead to unexpected behaviour

# Cache with Redis
# By default you should use something different than database 0 or 1, e.g. 13 (default max is 16, can be
configured in
# redis.conf) possible URLS are:
#    redis://[:password]@localhost:6379/0
#    rediss://[:password]@localhost:6379/0
#    unix://[:password]@/path/to/socket.sock?db=0
# CACHE_ENABLE: False
# CACHE_REDIS: False
# CACHE_REDIS_LOCATION: 'redis://127.0.0.1:6379/13'

# Disables Throttling (necessary for unittests to pass) by overriding the cache with a dummy cache
# https://docs.djangoproject.com/en/1.11/topics/cache/#dummy-caching-for-development
# THROTTLING: False

# Enables the management API, required for the psono-admin-client / admin portal
MANAGEMENT_ENABLED: True

# Allows that users can search for partial usernames
# ALLOW_USER_SEARCH_BY_USERNAME_PARTIAL: True

# Allows that users can search for email addresses too
# ALLOW_USER_SEARCH_BY_EMAIL: True

# Only necessary if the psono-client runs on a sub path (no trailing slash) e.g. "https://wwww.psono.pw"
# WEB_CLIENT_URL: ''

# Prevents the use of the last X passwords. 0 disables it.
# DISABLE_LAST_PASSWORDS: 0

# Your Postgres Database credentials
DATABASES:
   default:
      'ENGINE': 'django.db.backends.postgresql_psycopg2'
      'NAME': 'psono'
      'USER': 'psono'
      'PASSWORD': 'password'
      'HOST': 'psono-database'
      'PORT': '5432'
# for master / slave replication setup comment in the following (all reads will be redirected to the slave
#    slave:
#       'ENGINE': 'django.db.backends.postgresql_psycopg2'
#       'NAME': 'YourPostgresDatabase'
```

```
        #       'USER': 'YourPostgresUser'
        #       'PASSWORD': 'YourPostgresPassword'
        #       'HOST': 'YourPostgresHost'
        #       'PORT': 'YourPostgresPort'

        # The path to the template folder can be "shadowed" if required later
        TEMPLATES: [
          {
            'BACKEND': 'django.template.backends.django.DjangoTemplates',
            'DIRS': ['/root/psono/templates'],
            'APP_DIRS': True,
            'OPTIONS': {
              'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
              ],
            },
          },
]
```