THOMAS MORE

SOC MPI Oosterlo VZW

Realisatiedocument

Michiel Kuyken
Student Bachelor in de Elektronica-ICT – Cloud & Cyber Security

2024 - 2025

Inhoudsopgave

Overdrijf niet met indelingen en subindelingen. Werk liever niet met decimaal genummerde onderdeeltjes van slechts een tiental lijnen. Leer grotere gehelen schrijven die je logisch indeelt in alinea's.
Probeer je te beperken tot **3 niveaus**. Indien nodig kan je nog altijd indelen met behulp van bijvoorbeeld letters (a, b, c, …), opsommingstekens of cursief weergegeven kopjes, die uiteraard niet mee in de inhoudstafel worden opgenomen.
Update de inhoudsopgave pas nadat je volledig document uitgeschreven is.

1. Inleiding

Voor het laatste onderdeel van de opleiding Elektronica-ICT met afstudeerrichting Cloud & Cybersecurity, moesten we een stage afleggen bij een bedrijf of organisatie. Ik heb de kans gekregen om aan de slag te gaan bij het MPI Oosterlo VZW. In deze instelling ondersteunen ze jongeren en volwassenen met mentale beperkingen. Ze werken hier dus met gevoelige informatie over cliënten en het is net deze informatie die nuttig kan zijn voor aanvallers. Om dit te beveiligen hebben ze aan mij gevraagd om een SOC op te stellen.

Binnen het MPI Oosterlo was er tot nu toe nog geen centraal beveiligingssysteem dat op hun servers beveiligingsincidenten detecteert, analyseert en opvolgt. Hierdoor kunnen fouten snel door de vingers glippen en achterdeuren openblijven voor aanvallers. Dit brengt natuurlijk risico's mee op het vlak van data- en netwerkbeveiliging. De hoofdopdracht lag dan ook bij het voorzien van een centraal dashboard waarop incidenten weergegeven worden. Op dit dashboard moeten ze CVE's kunnen bekijken en hiervan meldingen krijgen op Teams. Er werd ook nadrukkelijk gevraagd om de mogelijkheden met Wazuh te bekijken en een mogelijke integratie met Crowdsec te voorzien. Het belangrijkste was om de kosten zo laag mogelijk te houden. Dit betekende dus zoveel mogelijk open-source tools gebruiken.

In de drie maanden die ik aan mijn SOC heb kunnen werken, hebben het team en ik meerdere malen de scope van het project moeten aanpassen. Met elke stap die we namen zijn we verder gaan verfijnen wat nuttig was voor de organisatie en wat niet. Uiteindelijk bleven we over met onderstaande tools en workflow.

Een alert wordt gedetecteerd met Wazuh en deze wordt naar Shuffle gestuurd, waar er vervolgens een workflow start. Deze workflow maakt alerts en cases aan in TheHive en voegt hier observables aan toe. Deze worden gecontroleerd door analyzers die zich in Cortex bevinden. Ten slotte krijgt het IT-team een melding in een Teams-chat die verteld wat er aan de hand is.

In dit document beschrijf ik voor u, de lezer, de volledige uitwerking van dit project. Ik begin met de analyse van de verschillende tools die ik gebruikt heb. Vervolgens bespreek ik de realisatie van het project, met alle overwinningen en obstakels die ik onderweg tegenkwam. In het slot haal ik aan welke toekomststappen MPI Oosterlo nog kan nemen om het project verder te optimaliseren.

Hier leid je je document in en refereer je naar de verschillende onderdelen. Dit document gaat over de eigenlijke realisatie, de uitvoering van je onderzoek of de oplossing van het probleem. Leg eerst de link met je projectplan en geef vervolgens de onderdelen die aan bod zullen komen in dit document.

2. Analyse

2.1. SIEM

2.1.1. Wazuh

a) Wat is Wazuh?

Wazuh is een open-source SIEM platform dat bedrijven en organisaties helpt bij het monitoren, detecteren en reageren op beveiligingsincidenten. Het doet dit door het verzamelen van logs van de verschillende endpoints in de IT-infrastructuur. Wazuh analyseert deze logs op verdachte activiteiten, zoals brute-force attacks of bekende kwetsbaarheden, en creëert hieruit alerts. Deze alerts worden geclassificeerd op basis van ernst. Een alert met lage ernst, zoals bijvoorbeeld een succesvolle aanmelding, krijgt een laag meldingsniveau toegewezen, terwijl een alert met hoge ernst, zoals een brute-force attack, een hoog meldingsniveau krijgt toegewezen. De meldingsniveaus gaan van 0 tot 15 en hier kan makkelijk op gefilterd worden. Aan de hand van deze niveaus kan een organisatie zelf bepalen welke alerts ze belangrijk vinden om bij te houden en welke niet.

Wazuh werkt onderliggend met drie componenten: de Wazuh Indexer, het Wazuh Dashboard en de Wazuh Server. Deze componenten kunnen samen worden geïnstalleerd op één VM, of apart op verschillende VM's. Het voordeel van de componenten samen te houden is dat dit het installatieproces vergemakkelijkt. Wazuh voorziet een script in hun documentatie waarmee je snel een werkend Wazuh-platform kunt opzetten.
Het nadeel hiervan is dat deze oplossing niet schaalbaar is. Voor een klein SOC kan dit geen probleem zijn, maar voor grote organisaties is het wel belangrijk om deze flexibiliteit te hebben. Daarom kunnen zij ervoor verkiezen om de componenten op aparte VM's te installeren. Op deze manier kunnen ze op lange termijn makkelijker rekening houden met een groeiend of krimpend aantal endpoints.

De Wazuh Indexer is verantwoordelijk voor het opslaan, doorzoekbaar maken en analyseren van alerts die gegenereerd zijn door de Wazuh Server. Het doet dit door de alerts te ontvangen van de Wazuh Server en deze op te slaan zodat ze later makkelijk doorzoekbaar zijn. Zo kun je als analist snel alerts filteren om bijvoorbeeld alle alerts van een specifieke endpoint te krijgen van de afgelopen 24 uur. De alerts kunnen ook makkelijk op het Dashboard zichtbaar gemaakt worden met verschillende soorten grafieken.
De Wazuh Indexer gebruikt achterliggend Opensearch, een open-source fork van het tegenwoordig betalende Elasticsearch. Elasticsearch is een zoek- en analysetool die instaat is om snel gestructureerde of ongestructureerde data te doorzoeken. Het staat Wazuh in staat om hun alerts te filteren en doorzoeken op basis van parameters zoals meldingsniveau of endpoint. Opensearch wordt door Wazuh gekozen zodat Wazuh alles zelf in handen heeft. Ze kiezen zelf welke richting ze uitwillen gaan en aangezien het open-source is, zijn er geen extra kosten. Wazuh kan nu ook hun Opensearch updates aanpassen aan die van de Wazuh Indexer om een soepele integratie tussen de twee te behouden.

De Wazuh Server is verantwoordelijk voor het beheren, configureren en updaten van de agents. Ze analyseren alle data die binnenkomt vanuit de agents en zoeken aan de hand van decoders, regels en threat intel naar IoC's. Deze worden dan gelabeld en opgeslagen in de Wazuh Indexer voor verdere analyse. De verschillende alerts zijn ook zichtbaar op het Dashboard. Via de Wazuh manager is het ook mogelijk om verschillende aanpassingen aan de configuratie te maken. Zo kun je instellen vanaf welke meldingsniveau je alerts wil beginnen genereren, op maat gemaakte filters maken die alerts filteren, connecties maken met andere tools zoals Shuffle en TheHive, … Op deze manier zorgt Wazuh ervoor dat er veel flexibiliteit is voor zijn gebruikers en kan iedereen zijn Wazuh instellen naar zijn noden.

Het Wazuh Dashboard is de web interface die de alerts visualiseert voor de gebruiker. Dit gebeurt voornamelijk aan de hand van grafieken. Het Dashboard is ook een plaats om aanpassingen te maken aan de configuratie. Dit kan doormiddel van verschillende tools op het Dashboard zoals DevTools. Hiermee kunnen er API-calls gedaan worden naar de Wazuh Indexer waarmee data opgehaald, verwijdert of aangepast kan worden. In Bijlage x staat een handleiding die ik geschreven heb om alerts te verwijderen per dag met behulp van DevTools.

b) Mogelijkheden met Wazuh?

Wazuh biedt tal van extra mogelijkheden voor verdere configuratie dan enkel het maken en opslaan van alerts. Hieronder worden een aantal mogelijkheden gegeven die ook worden bekeken in het kader van deze stageopdracht.

Security Configuration Assessement (SCA)

Wazuh monitort de systeem- en applicatieconfiguratie om te controleren of deze in orde zijn. De controle hiervan gebeurt met behulp van CIS-benchmarks of andere hardening guides. Deze tools controleren een endpoint om te kijken hoe veilig deze is. Dit wordt weergegeven op basis van een score, des te hoger de score des te beter. Wazuh agents voeren periodieke scans uit waarin ze opzoek gaan naar gaten in de beveiliging of slechte configuraties waar misbruik van gemaakt kan worden en passen de score aan. De resultaten van de scans worden getoond op het Dashboard.

File Integrity Monitoring (FIM)

Wazuh monitort het filesysteem van de agent op alle veranderingen die gebeuren. Dit kan gaan van een aanpassingen in de contents van een bestand tot het veranderen van permissies op bestanden. Wazuh kan op deze manier schadelijke files detecteren en gecompromitteerde endpoints identificeren.

Threat hunting

Wazuh onderzoekt bepaald gedrag op endpoints dat kan duiden op een IoC. Het doet dit met behulp van het MITRE ATT&CK framework om veelvoorkomende TTP's op te sporen. Met behulp van threat hunting worden risico's opgespoord die voorbij de eerste securitylaag geraakt zijn.

Vulnerability detection

Wazuh controleert software die draait op endpoints voor mogelijke CVE's die zich hierop bevinden. CVE's zijn common vulnerabilities and exposures die bijgehouden worden in verschillende databases. Op deze manier kunnen andere mensen die dezelfde kwetsbaarheden op hun systemen hebben deze databases raadplegen en een oplossing vinden. Wazuh deelt de CVE's in, op basis van de score die ze krijgen, onder Low, Medium, High en Critical.

Incident response

Wazuh biedt ook de mogelijkheid om meteen tegen de alerts en threats in te gaan met hun Incident Response. De Incident Response schiet in actie als er aan bepaalde eisen voldaan worden. Een voorbeeld van een Response-techniek is om het endpoint volledig af te sluiten van het netwerk. Op deze manier is er geen dreiging voor andere systemen in hetzelfde netwerk en kan de threat ook niet meer verbinden met ons systeem.

Regulatory compliance

Tegenwoordig zijn er ook veel verplichtingen vanuit de wetgeving waar organisaties zich aan moeten houden. In de EU is er de GDPR-wetgeving die organisaties zegt hoe ze met gegevens moeten omgaan. Wazuh helpt bedrijven door te controleren of ze aan deze wetgeving voldoen door middel van FIM en SCA. Wazuh biedt ook ondersteuning voor andere wetgevingen, die voor deze opdracht minder belangrijk zijn.

IT Hygiene

In IT is een geoptimaliseerd systeem nodig om alles soepel te laten werken en alles veilig te houden. Onnodige open poorten kunnen leiden tot backdoors voor hackers en dit moet vermeden worden. Wazuh biedt daarom de mogelijkheid om van alle endpoints de lopende applicaties, open poorten en informatie over het OS-systeem en hardware bij te houden. Zo kan er op een centrale plaats gekeken worden naar de aparte endpoints en kan er op een visuele manier verbeteringen gespot worden.

c) Waarom Wazuh?

Voor mijn stageopdracht ga ik gebruik maken van Wazuh als SIEM in mijn SOC. Vanuit mijn stageplaats kwam al de interesse om deze tool te gebruiken en dit stond ook in de stageopdracht vermeld. Ze hadden dit zelf al eens uitgetest en wouden graag de verdere mogelijkheden bekijken dat Wazuh te bieden heeft. De Dashboard functie die Wazuh aanbiedt gaat ook zeker van pas komen. In mijn stageopdracht staat ook vermeld dat ze graag een centraal dashboard willen en dat van Wazuh zelf is heel duidelijk en voldoet aan al hun eisen. Het gaat ook de overdracht makkelijker maken, omdat alles wat via CLI aangepast kan worden ook in het dashboard kan gebeuren. Hier ga ik zeker rekening mee houden tijdens de realisatie.

Een andere drijfveer is dat Wazuh een open-source platform is. Het MPI Oosterlo is een VZW waardoor er gestreefd wordt om geen onnodige kosten te maken. Er werd ook duidelijk vermeld aan het begin van mijn stage dat de voorkeur altijd uitgaat naar open-source oplossingen die lokaal gehost kunnen worden. Wazuh ligt helemaal in lijn met wat ze op dit vlak in gedachten hebben en past daarom perfect in mijn SOC.

Toen ik de opdracht las om een SOC te maken voor het MPI Oosterlo, was standaard mijn voorkeur al om Wazuh als SIEM te gaan gebruiken. Het is een tool die we tijdens de lessen gezien heb en die ik al eens eerder heb gebruikt. Ik voel me er dus best comfortabel bij om dit op te zetten in hun omgeving, zonder me zorgen te moeten maken dat er iets fout gaat gaan. Aangezien het SIEM het startpunt is van mijn SOC, leek het me best om met een vertrouwde tool te beginnen zodat ik eerst wat zelfvertrouwen opbouw binnen de organisatie, alvorens nieuwe tools te gaan gebruiken.

Tenslotte kwam de voorkeur ook door Wazuh dat gewoon een heel goed platform is. De online documentatie op hun site is heel duidelijk en er is veel te vinden als ik ergens vastloop. Wazuh voorziet een ganse pagina voor enkel troubleshooting die ik kan raadplegen zouden er ergens problemen zijn. Ook de online community is groot en op Reddit of Discord kun je altijd bij hen terecht voor vragen of staan er links naar YouTube video’s waar het wordt uitgelegd.

d) WRM

2.2. SOAR

2.2.1. Shuffle

a) Wat is Shuffle?

Shuffle is een open-source SOAR-platform dat ontworpen is om een groot probleem in de cybersecuritywereld aan te pakken: er moet te veel handmatig gebeuren. In de wereld van cybersecurity draait alles om snel en op een gestructureerde manier te reageren op aanvallen of andere bedreigingen. Sommige organisaties proberen dit allemaal handmatig te doen door eerst zelf op zoek te gaan naar de bedreiging, hierna de bedreiging zelf te onderzoeken en ten slotte zelf op zoek te gaan naar een oplossing. Deze tactiek zorgt voor veel saai en repetitief werk dat de aandacht van de belangrijke incidenten weghaalt. Het is voor grote organisaties ook niet haalbaar om deze manier toe te passen. Daarom proberen zij één grote tool te maken die alles doet van het opsporen van de bedreiging, tot het onderzoeken ervan, een case maken en de bedreiging oplossen. Dit zijn zware platformen die veel tijd kosten om te maken en die bedrijven graag voor zichzelf houden. Shuffle probeert hiervan af te stappen door een platform aan te bieden waar dit allemaal gedaan kan worden, zonder zelf een zwaar platform te moeten maken. Hun missie is om met hun platform processen en workflows onderling te delen, een gestandaardiseerde aanpak van detecties en automatiseringen creëren en samenwerkingen tussen organisaties bevorderen. Zoals de eigenaar ook zegt: "*Cybersecurity is not a competition, and shouldn't be treated as such.*".

Shuffle integreert verschillende andere tools en platformen door gebruik te maken van hun API's. De meeste platformen voorzien deze API's zodat organisaties tijdrovende taken kunnen automatiseren. Shuffle zet nog een stap verder door een gebruiksvriendelijke web interface te maken, waaruit deze API-calls gemaakt kunnen worden naar de platformen. Hoewel de gebruiker zelf de platformen nog moet opzetten, als deze niet in de cloud beschikbaar zijn, bespaart het veel tijd met het uitzoeken hoe de API werkt en zelf de calls te maken en automatiseren. Shuffle voorziet een integratie met heel veel verschillende platformen en tools waardoor het voor iedereen beschikbaar is.

Zoals net al vermeld werkt Shuffle met een web-interface die het gebruik van Shuffle makkelijker maakt. Hierachter ligt een grote API die gebruikt wordt om de andere API's aan te spreken. Het is ook mogelijk om deze API zelf te gebruiken. Ongeacht welke optie gekozen wordt, staat hier security natuurlijk ook centraal. Shuffle maakt gebruik van Bearer Auth om de API-calls te beveiligen. Dit houdt in dat er bij elke request een API-key wordt meegestuurd die aangeeft dat jij de request naar de API maakt. Zo moet er niet constant een gebruikersnaam en wachtwoord meegestuurd worden. De token zorgt dus voor goede security van de API, maar behoudt toch zijn flexibiliteit om gebruikt te worden in scripts of andere automatiseringstools.

Om Shuffle te gebruiken zijn er enkele mogelijkheden beschikbaar. Je kunt alles zelf lokaal hosten, gebruik maken van de cloud versie of een mix van de twee en kiezen voor een hybride oplossing. Elk heeft zijn voordelen zodat er per organisatie gekeken kan worden wat het beste voor hun past. Zo heb je bij een lokale hosting meer controle over de infrastructuur, maar moet je dan ook alles zelf up-to-date houden. Dit is een goede oplossing voor kleinere organisaties of VZW's.
In de cloud geef je dit dan weer uit handen zodat je minder onderhoud hebt en makkelijker kunt schalen, maar hier kunnen wel kosten aan verbonden zijn. Er zijn limieten gezet op de gratis versie van de cloud, zoals maximaal 10 workflows in totaal of 2000 App-runs per maand. Je kunt deze limieten vergroten door een abonnement te nemen. Deze abonnementen kunnen vaste limieten hebben of op basis van je organisatie gemaakt worden.
Een hybride oplossing klinkt dan als het beste van twee werelden, maar de set-up is complexer en je hebt veel kennis nodig van zowel lokaal als cloud-hosting om dit goed te laten werken.
Kortom is er voor iedere organisatie wel iets en is dit een extra laag flexibiliteit die Shuffle aanbiedt.

b) Mogelijkheden met Shuffle?

Om Shuffle zijn automatisaties en flexibiliteit te geven, zijn er verschillende features toegevoegd om dit makkelijker te maken voor de eindgebruiker. Hieronder kunt u de meest gebruikte features vinden die ik in mijn SOC ga gebruiken.

Workflows

Workflows zijn de kern van Shuffle en zorgen ervoor dat de dagelijkse taken geautomatiseerd worden. Ze bestaan uit apps, triggers, voorwaarden en variabelen om een krachtige en gebruiksvriendelijke automatiseringen op te zetten.
Shuffle staat toe dat je meerdere workflows kunt maken die elk hun eigen doeleinde hebben. Zo kun je een workflow maken die zich puur richt op het maken van cases in TheHive vanuit Wazuh alerts en een andere workflow die zich focust op het analyseren van mails op phishing. Verschillende workflows kunnen op hetzelfde moment lopen, waardoor er meer taken gedaan kunnen worden op minder tijd. Ze kunnen ook automatisch gestart worden op basis van een webhook of een geplande taak.
Shuffle biedt ook verschillende templates aan die gemaakt zijn door andere gebruikers. Zo hoef je niet alle workflows zelf te maken en kun je inspiratie opdoen van wat anderen gemaakt hebben.

Apps (openAPI or self-made in python)

Apps zijn de bouwstenen waaruit workflows opgebouwd worden. Het is een integratie met een externe tool, zoals Wazuh, TheHive, Outlook, VirusTotal … die gebruikt kunnen worden om verschillende acties uit te voeren op de tool. Deze acties kunnen bestaan uit het ophalen van alerts uit Wazuh, het aanmaken van een case in TheHive, een analyse starten met VirusTotal, … De acties worden vooraf gedefinieerd door de maker van de app en komen overeen met de functies die de externe tool aanbiedt.
Om ervoor te zorgen dat de actie succesvol uitgevoerd wordt, zijn er argumenten nodig. Deze argumenten kunnen vergeleken worden met variabelen waarin je bepaalde gegevens meegeeft zoals een IP-adres dat onderzocht moet worden, of een alert waarvan we een case willen maken. Elke actie heeft enkele verplichte en optionele argumenten die bepaald worden door de maker van de app.
Apps worden dus vooral gebruikt om automatische acties uit te voeren op de externe tools. Standaard voorziet Shuffle meer dan 2500 apps, maar je kunt er ook zelf maken in Python.

Triggers

Automatisatie zou geen automatisatie zijn als er nog steeds zaken handmatig moeten gebeuren, zoals bijvoorbeeld het starten van een workflow. Daarom voorziet Shuffle verschillende triggers die een workflow kunnen starten. Er zijn in totaal vier soorten triggers die veel voorkomen: webhooks, schedules, subflows en user input.
Een webhook luistert naar inkomende data van andere tools, zoals een alert op Wazuh. Zodra die tool iets nieuws wil melden, schiet de rest van de workflow in gang om met de nieuwe data aan de slag te gaan. Zo kan er bijvoorbeeld een case gemaakt worden in TheHive van de Wazuh alert.
Een schedule is een taak die op een bepaald tijdstip de workflow aan zet. Dit kan gebruikt worden om elke dag nieuwe IOC's op te halen uit MISP en hier een lijst van opsturen via mail.
Een subflow is een trigger van een workflow naar een andere workflow. Dit kan gebruikt worden bij het analyseren van een IP-adres. Je kunt telkens verschillende acties toevoegen die het IP-adres controleert, maar je kunt ook een workflow maken die deze acties ook doet en waarnaar verwezen wordt vanuit een andere workflow.
Ten slotte is er nog user input. Dit is wanneer een gebruiker een bepaalde handeling uitvoert, zoals het downloaden van een bestand, dat een workflow aanzet om te controleren of het bestand wel veilig is.

Marketplace

Om alle bovenstaande zaken makkelijk te kunnen vinden en niet zelf alles handmatig te moeten maken, voorziet Shuffle een marketplace. Op deze marketplace kun je vanalles vinden zoals nieuwe apps en bestaande workflows. Shuffle vindt het belangrijk om dit soort informatie met elkaar te delen en daarom is een plaats zoals de marketplace net extra belangrijk om elkaar te helpen met het beveiligen van je organisatie.

c) Waarom Shuffle?

Voor mijn stageopdracht ga ik gebruik maken van Shuffle omwille van de automatisering die het biedt voor mijn SOC. Één van de grote voordelen van een SOC is dat er veel zaken automatisch op een centrale plaats gebeuren. Dit bespaart veel tijd die anders verloren gaat aan repetitieve taken. Omdat het MPI Oosterlo een kleine IT-dienst heeft, is deze automatisatie van cruciaal belang. Ze zijn namelijk niet enkel bezig met het onderzoeken van mogelijke bedreigingen, ze moeten ook het personeel ondersteunen als zij problemen ondervinden. Het is dus belangrijk dat als ze gebruik willen maken van mijn SOC, dat alles voor hen klaar staat om op mogelijke bedreigingen in te spelen.

Shuffle is, net als Wazuh, een open source tool die volledig lokaal gehost kan worden. De voorkeur ging al uit naar dit soort tools, zodat ze geen overbodige kosten maken en zelf veel controle hebben over de tool.

Om centraal beheer van mijn SOC te behouden, is Shuffle een centrale plaats om alle API-verbindingen te zien. Hoewel deze integraties ook in de andere tools zelf zit, zoals tussen Wazuh en TheHive, is het moeilijk om een overzicht hierover te houden. Met Shuffle kunnen we alle API-integraties en taken die automatisch uitgevoerd worden zien. Dit zorgt voor een centrale plaats waar we makkelijk aan troubleshooting kunnen doen, mochten er fouten optreden.

Tenslotte is Shuffle een tool waarmee ik al eens gewerkt heb. Dit zorgt ervoor dat ik al een basis heb om mee van start te gaan.

d) WRM

2.2.2. TheHive

a) Wat is TheHive?

TheHive is een open-sourceplatform voor security incident en response dat essentieel is voor beveiligingsteams zoals SOC-, CSIRT- en CERT-teams. Het stelt hen in staat om incidenten snel en doeltreffend af te handelen. Het biedt een platform aan waarop teams makkelijk kunnen samenwerken aan verschillende incidenten en hier analyse op uitvoeren. Door de vlotte integratie met andere beveiligingstools en de schaalbaarheid en flexibiliteit die het platform biedt, is het inzetbaar in zowel grote als kleine omgevingen. Dankzij de gebruiksvriendelijke interface worden de incidenten op een overzichtelijke manier bijgehouden en worden analyses uitvoeren makkelijker.

In TheHive werkt met organisaties om de incidenten naar op te sturen. Deze organisaties worden aangemaakt in het admin-portaal, waar het account beschikbaar voor wordt gemaakt als TheHive opgezet is. De organisaties kunnen voor verschillende doeleinden gebruikt worden. Bij kleinere omgevingen kan er slechts één organisatie opgezet worden waar alle incidenten bijgehouden worden. Bij grotere omgevingen is het voordeliger om verschillende organisaties op te zetten voor de verschillende analyse teams. Zo kunnen incidenten overzichtelijk blijven en moeten teams niet zoeken naar incidenten waar zij verantwoordelijk voor zijn.

Incidenten worden geplaatst in de verschillende organisaties. Dit kan handmatig gebeuren in TheHive zelf of automatisch door gebruik te maken van de API. De incidenten worden bewaard onder de tab “alerts” waarin de uitleg over het incident vermeld staat.
In deze alerts kan belangrijke informatie, zoals IP-adressen, domeinnamen en bestandsnamen, opgeslagen worden in observables. Met deze observable kan er een case gemaakt worden waarin alle informatie gebundeld wordt. Door de integratie van TheHive met beveiligingstools zoals VirusTotal, kan er een automatische analyse gestart worden van het incident. Dit bespaart de gebruikers binnen de organisatie veel tijd om handmatig analyses uit te voeren en kunnen ze hun focus leggen op de response.

Om analyse uit te voeren in de organisaties, zijn er gebruikers nodig die hier toegang tot hebben. Vanuit het admin-portaal is er de mogelijkheid om gebruikers toe te voegen aan een organisatie, zolang er hiervoor een email-adres wordt meegegeven. Op deze manier kan elke organisatie zijn eigen gebruikers beheren en kunnen zij enkel voor hun relevante cases zien. In TheHive gebruiken ze RBAC om de juiste rollen toe te kennen aan gebruikers. Deze rollen zijn:

- Admin:
  - Volledig administratieve rechten
  - Enkel op admin-portaal
  - Kan geen cases en alerts bekijken
- Org-admin:
  - Beheert gebruikers en configuraties binnen de organisatie
  - Heeft een API-key voor integratie met andere tools
  - Kan cases en alerts aanmaken en bekijken
  - Kan analyzers en responders starten
- Analist
  - Kan cases en alerts aanmaken en bekijken
  - Kan analyzers en responders starten
- Read-only
  - Kan cases en alerts bekijken

Organisaties kunnen hiermee bepalen hoe ze hun teams willen opstellen in TheHive en makkelijk de juiste rechten toekennen.

TheHive werkt met een licentiesysteem, ongeacht of je de community versie of een betalende versie gebruikt. Je moet je als organisatie aanmelden om in aanmerking te komen voor een licentie. Met deze licentie kun je één van de drie versies van TheHive aanvragen: community, gold of platinum. Elk hebben hun eigen voordelen, maar de betalende versies kunnen oplopen tot €25.000 voor on premise of zelfs €50.000 voor in de cloud.

b) Mogelijkheden met TheHive

TheHive biedt tal van mogelijkheden

Integration with MISP
Real-time collaboration
Efficient task management
Customizable templates
Evidence management
Observable management
Threat intelligence integration

c) Waarom TheHive?

d) WRM

2.2.3. Cortex

a) Wat is Cortex?

b) Mogelijkheden met Cortex?
c) Waarom Cortex?
d) WRM

2.2.4. Teams

a) Wat is Teams?
b) Mogelijkheden met Teams?
c) Waarom Teams?
d) WRM

2.3. Threat Intel

2.3.1. VirusTotal

a) Wat is VirusTotal?

VirusTotal is een gratis online platform dat parameters analyseert door informatie op te halen van meer dan 90 verschillende beveiligingstools, zoals antivirussen en blocklists. De gebruiker krijgt een score te zien die verteld hoeveel van de beveilingstools hun parameters beschouwen als schadelijk.
Naast deze score wordt er extra informatie ter beschikking gesteld zoals:

- Meer informatie over de beveiligingstool
- De categorisatie van de dreiging, zoals malware, phishing, botnet, …
- Meer informatie over deze categorieën en wat de risico's hiervan zijn

VirusTotal helpt gebruikers bij het opsporen van malware en false positives om bij te dragen aan de algemene veiligheid in het IT-landschap. Centraal staat de real-time updates dat ervoor zorgt dat ze altijd up to date zijn met de recentste veranderingen, zoals nieuwe schadelijke IP's of domeinen.

Er zijn verschillende manieren om een analyse te starten met VirusTotal. Eerst en vooral is er de website die het meest gebruikt wordt. Op deze website is er een zoekbalk waarin de gebruiker de parameters kan ingeven of uploaden zodat ze geanalyseerd worden.
Er is ook een browserextensie die webpagina's scant voor malware of virussen en bestanden die gedownload worden. Het geeft dan, net zoals bij de website, een score die aanduidt hoe veilig een pagina of bestand is.
Ten slotte is er nog een API. Deze is interessant voor automatische analyses uit te voeren door een request naar de API te sturen. Er wordt dan opnieuw een score gegeven zoals bij de andere opties, maar nu in JSON-formaat.

Om valse positieven te voorkomen, heeft VirusTotal de VirusTotal-community opgericht. In deze community wordt er gekeken naar de resultaten die gegeven worden voor de parameters. Als er fouten worden opgespoord of nieuwe dreigingen worden ontdekt, kan de community helpen met deze op te sporen en aan te geven.

b) Mogelijkheden met Virustotal?

Virustotal onderzoekt verschillende soorten parameters. Hieronder gaan we dieper op in welke parameters dit zijn en wat de gebruiker van informatie krijgt.

IP-adressen

Een belangrijke parameter om te onderzoeken zijn IP-adressen. Online zijn er veel lijsten met bekende schadelijke IP-adressen en IP's die gebruikt worden in botnets. VirusTotal heeft een integratie met deze lijsten zodat er automatisch verteld wordt of het IP in een lijst te vinden is. Natuurlijk is dit niet het enige wat een IP gevaarlijk kan maken voor een organisatie. Daarom laat VirusTotal historische informatie zien over activiteiten gekoppeld aan het IP-adres. Deze activiteiten bevatten:

- Bestanden die gehost zijn op het IP-adres
- Domeinen die gekoppeld zijn aan het IP-adres
- Gedetecteerde kwaadaardige activiteiten, zoals Command & Control servers

Sommige tools geven een reputatiescore zoals malicious, suspicious of clean. Dit kan helpen om mensen aan te sporen verder onderzoek te doen naar het IP-adres.
Virustotal geeft ook enkele belangrijke metadata mee. Dit kan het ASN bevatten dat het indentificatienummer is van het netwerk waarin het IP zich bevindt. Soms wordt ook de geolocatie van het IP bekend gemaakt. Dit is handig voor organisaties die willen achterhalen uit welke landen de meeste aanvallen komen, zodat ze deze landen eventueel kunnen blokkeren.

URL's

Eén van de grootste beveiligingsrisico's tegenwoordig is phishing. Vaak bevatten mails een link die naar een schadelijke website, die een bekende website nabootst, waar er gevraagd wordt om gegevens in te vullen. VirusTotal controleert URL's om te kijken of deze veilig zijn. Het scant de URL met tientallen website-scanners en blocklistingsdiensten zoals Google Safe Browsing, Fortinet, Sophos, …
De scanresultaten bevatten informatie zoals:

- Is de site phishing-gerelateerd?
- Malware die gehost wordt op de site
- Classificatie van de site (verdacht, spam)

Het toont ook andere informatie zoals doorverwijzingen die de URL uitvoert. Door een doorverwijzingen te doen kan de URL als veilig beschouwd worden, maar is de doorverwijzingen naar een URL die onveilig is. Door beide te controleren kan bepaald worden of de URL helemaal veilig is. De relatie die de URL heeft met andere domeinen, bestanden en IP-adressen wordt ook gecontroleerd en getoond aan de gebruiker voor extra veiligheid te bieden.

File hashes

Een file hash is de unieke identificatie van een bestand. Het is een onomkeerbaar identificatienummer waardoor bestanden online herkent kunnen worden. VirusTotal controleert deze hash om te kijken of het bestand ergens gebruikt wordt voor slechte doeleinden en of er malware aan gelinkt is. Bij het opzoeken van een hash op VirusTotal krijg je informatie over het bestand zoals:

- Eerdere uploads of scans van het bestand op VirusTotal
- Een volledig scanrapport van de antivirusprogramma's en welke het bestand als schadelijk detecteren
- Extra details zoals het bestandstype, de grootte, metadata die bij in het bestand zit, …

VirusTotal linkt het bestand ook aan URL's, IP-adressen, downloadlocaties en gerelateerde samples om verdere detectie makkelijker te maken.

Domeinen

Domeinen worden gebruikt om te verwijzen naar een specifiek adres op het internet, zoals microsoft.com. Deze adressen kunnen gebruikt worden om aanvallen te starten of malware te verspreiden. Dit gebeurt vaan door een domein na te maken, zoals rnicrosoft.com, waar mensen niet nauwkeurig genoeg naar kijken. Ze proberen de website dan zo realistisch mogelijk na te maken om gebruikers malware te laten downloaden of gevoelige data te delen. Domeinen worden ook vaak gebruikt in phishingmails waarin ze een email sturen vanuit hun nepdomein dat lijkt op een officiële mail van Microsoft.
VirusTotal verzamelt de volgende gegevens en informatie van domeinen om ze aan hun gebruikers te laten zien:

- IP-adressen gelinkt met het domein
- Geregistreerde subdomeinen
- Malware- of phishing-hosting
- Reputatiescores

Voor verdere analyse is het ook mogelijk om de WHOIS-gegevens te krijgen van het domein. Dit is informatie zoals wie het domein heeft geregistreerd, wanneer het geregistreerd is, wanneer het verloopt, waar het geregistreerd is, … Aan de hand van deze informatie kan er dieper worden gecontroleerd of een domein onveilig is voor een organisatie of niet.
Het is ook mogelijk om de SSL-certificaten van het domein te bekijken. Een site die HTTPS gebruikt, betekent niet altijd dat deze veilig is. Een certificaat voor HTTPS kan komen van een onofficiële site zoals Let's Encrypt om mensen een vals gevoel van veiligheid te geven. Daarom is het belangrijk om te kunnen kijken van waar het certificaat komt om te bepalen of het domein veilig is.
Ten slotte is het ook mogelijk om verbonden bestanden en URL's van het domein te bekijken en deze te onderzoeken of ze veilig zijn.

c) Waarom VirusTotal?

2.3.2. Crowdsec

a) Wat is Crowdsec?
b) Mogelijkheden met Crowdsec?
c) Waarom Crowdsec?

Je geeft de analyse, de gebruikte tools en waarom je hiervoor gekozen hebt. Kortom wat heb je onderzocht… ter voorbereiding van de werkelijke uitvoering van je opdracht?
Denk eraan dat je bijvoorbeeld een vergelijking maakt van mogelijke ontwikkelomgevingen, platformen, technologieën, enz., en daarna verantwoorden met de Weighted Ranking Methode waarom je bepaalde keuzes gemaakt hebt. Zelfs als de keuze vooraf al vastlag, is het interessant dat je deze analyse in je realisatiedocument vermeldt en toelicht.

Denk eraan dat ook op hoofdstukniveau een goede inleiding en afsluiting erg waardevol kunnen zijn. Het is bijvoorbeeld geen goed idee om als volgt te werk te gaan:

1 Gebruikte tools
1.1 EclIP’se

Het spreekt eigenlijk voor zich dat je het onderdeel "Gebruikte tools" eerst kort inleidt (ev. mét een overzicht), en dat je pas daarna de verschillende tools in detail gaat behandelen.

3. SOC-realisatie

3.1. SIEM

3.1.1. Wazuh

SETUP

Zoals eerder vermeld bestaat Wazuh uit drie componenten: de Wazuh Indexer, de Wazuh Server en het Wazuh Dashboard. Deze componenten zijn essentieel om Wazuh goed te laten werken. Om de componenten samen te installeren, zijn er twee mogelijkheden die Wazuh ons aanbiedt. Enerzijds kunnen we alles op één VM plaatsen om het installatieproces makkelijker te maken. Dit gaat dan wel ten koste van toekomstige schaalbaarheid. Anderzijds kunnen de componenten op drie verschillende VM's geïnstalleerd worden. Dit kost meer resources en kan lastiger zijn, maar biedt de mogelijkheid om verder te schalen op lange termijn. Ik heb beide opties geprobeerd, maar omdat mijn SOC zich in een dynamische omgeving bevindt waar regelmatig veranderingen plaatsvinden, leek het mij het beste om de laatste optie te kiezen. Hierdoor vorm ik niet alleen een oplossing die ze de komende tijd kunnen gebruiken, maar ook een die ze kunnen blijven aanpassen aan toekomstige noden. Momenteel is alles met single-node geïnstalleerd, wat betekent dat er slecht één van elke component is. Er was nog geen behoefte om uit te breiden naar een multi-node cluster, omdat dit meer resources in beslag zou nemen.

Wazuh Indexer

a) Installatie

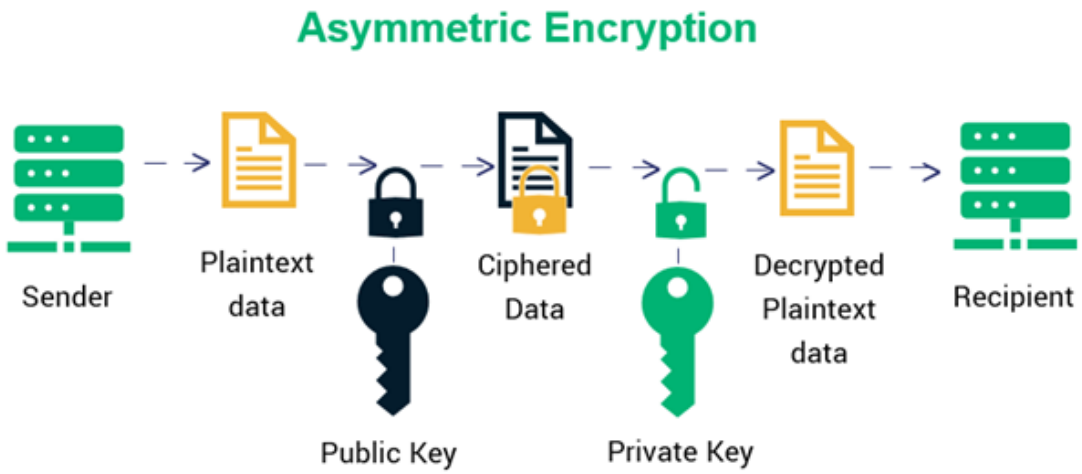
De installatiegids van Wazuh begint met installeren van de Wazuh Indexer. Dit is het centrale component waarop de alerts worden opgeslagen. Er wordt met de Indexer begonnen omdat hier het 'config.yml' bestand wordt opgeslagen, dat u hieronder kunt zien. Dit bestand bevat alle IP's van de verschillende componenten die worden opgezet. Zo weet Wazuh welke verbindingen er opgezet worden. Ook wordt hier de name van de nodes bewaard die we later gaan nodig hebben bij de configuratie van de componenten.

```yaml
nodes:
  # Wazuh indexer nodes
  indexer:
    - name: node-1
      ip: "172.17.0.232"
    #- name: node-2
    #  ip: "<indexer-node-ip>"
    #- name: node-3
    #  ip: "<indexer-node-ip>"

  # Wazuh server nodes
  # If there is more than one Wazuh server
  # node, each one must have a node_type
  server:
    - name: wazuh-1
      ip: "172.17.0.225"
    #  node_type: master
    #- name: wazuh-2
    #  ip: "<wazuh-manager-ip>"
    #  node_type: worker
    #- name: wazuh-3
    #  ip: "<wazuh-manager-ip>"
    #  node_type: worker

  # Wazuh dashboard nodes
  dashboard:
    - name: dashboard
      ip: "172.17.0.233"
```

Config.yml bestand uit mijn SOC

Tussen deze componenten wordt veel gevoelige informatie gedeeld. Wazuh voorziet daarom een script waarmee certificaten gegenereerd worden om de communicatie te encrypteren. De certificaten moeten hierna nog gecomprimeerd worden zodat ze makkelijk gedeeld kunnen worden tussen de verschillende VM's met het scp commando.

Voordat de Wazuh Indexer geïnstalleerd kan worden, moeten we eerst een GPG-sleutel voorzien. Dit is een sleutelpaar dat bestaat uit een private en publieke sleutel. Dit sleutelpaar controleert softwarepakketten om na te kijken of ze echt afkomstig zijn van de maker en niet meer aangepast zijn door een derde partij. Het garandeert dat de bestanden niet corrupt of gemanipuleerd zijn tijdens de overdracht.
Wazuh maakt gebruik van asymmetrische encryptie. Hierbij wordt data versleuteld met een publieke sleutel en kan deze alleen worden ontsleuteld met de bijbehorende private sleutel, die Wazuh in bezit heeft. Op de figuur hieronder kunt u zien hoe dit in zijn werk gaat. Omdat de data enkel ontcijferd kan worden als er niks aangepast is, weten we dat de integriteit niet geschonden is en dat het bestand veilig is. Als er bij het decrypteren fouten opduiken, wordt er een waarschuwing uitgestuurd dat het bestand waarschijnlijk niet meer veilig is. Door asymmetrische encryptie te gebruiken, kan Wazuh voorkomen dat iemand een Man-in-the-middle aanval opzet tussen de Wazuh Server en Wazuh Indexer. Mocht iemand erin slagen om dit te doen, kunnen ze alle alerts eerst langs hen laten komen waardoor ze veel inzicht krijgen in een organisatie en een mogelijke achterdeur kunnen vinden.



Proces asymmetrische encryptie

Nu dat we de integriteit kunnen controleren, is het tijd om de Wazuh Indexer te installeren op de VM. Wazuh voorziet hiervoor een package die makkelijk te installeren is. In deze package zit alles wat nodig is om te kunnen beginnen aan de configuratie van de Wazuh Indexer, zoals Opensearch en de configuratiebestanden.

b) Configuratie

Nu alle nodige bestanden en software geïnstalleerd is, kunnen we beginnen met het configureren van de Indexer. De eerste aanpassingen gebeuren in het configuratiebestand van Opensearch, namelijk ‘opensearch.yml’, dat hieronder weergeven wordt. Het eerste wat aangepast wordt, is de ‘network.host’. Dit bevat het IP-adres of de hostnaam waarnaar er geluisterd moet worden voor HTTP-verkeer alsook intern transportverkeer. Dit adres of deze hostnaam worden gebruikt om verbindingen te accepteren en om aan andere servers te laten weten waarmee de node samenwerkt.
Het volgende dat aangepast wordt is de ‘node.name’. Dit bevat de naam van de node zoals het ook aangegeven staat in de config.yml. Als deze naam hiervan afwijkt, zal Wazuh de node niet herkennen en kan er geen verbinding opgezet worden.
Het laatste dat nog aangepast moet worden is de ‘cluster.initial_master_nodes’. Hieronder staan de namen van alle nodes die zich in een cluster bevinden. In ons geval gaat dit slechts één node zijn, omdat we werken met een single-node installatie. Als er gebruik gemaakt wordt van een cluster, worden hier de namen van de andere nodes ingegeven zoals ze ook opgegeven staan in de config.yml.

```
network.host: "172.17.0.232"
node.name: "node-1"
cluster.initial_master_nodes:
- "node-1"
#- "node-2"
#- "node-3"
cluster.name: "wazuh-cluster"
#discovery.seed_hosts:
#  - "node-1-ip"
#  - "node-2-ip"
#  - "node-3-ip"
node.max_local_storage_nodes: "3"
path.data: /var/lib/wazuh-indexer
path.logs: /var/log/wazuh-indexer

plugins.security.ssl.http.pemcert_filepath: /etc/wazuh-indexer/certs/indexer.pem
plugins.security.ssl.http.pemkey_filepath: /etc/wazuh-indexer/certs/indexer-key.pem
plugins.security.ssl.http.pemtrustedcas_filepath: /etc/wazuh-indexer/certs/root-ca.pem
plugins.security.ssl.transport.pemcert_filepath: /etc/wazuh-indexer/certs/indexer.pem
plugins.security.ssl.transport.pemkey_filepath: /etc/wazuh-indexer/certs/indexer-key.pem
plugins.security.ssl.transport.pemtrustedcas_filepath: /etc/wazuh-indexer/certs/root-ca.pem
plugins.security.ssl.http.enabled: true
plugins.security.ssl.transport.enforce_hostname_verification: false
plugins.security.ssl.transport.resolve_hostname: false

plugins.security.authcz.admin_dn:
- "CN=admin,OU=Wazuh,O=Wazuh,L=California,C=US"
plugins.security.check_snapshot_restore_write_privileges: true
plugins.security.enable_snapshot_restore_privilege: true
plugins.security.nodes_dn:
- "CN=node-1,OU=Wazuh,O=Wazuh,L=California,C=US"
#- "CN=node-2,OU=Wazuh,O=Wazuh,L=California,C=US"
#- "CN=node-3,OU=Wazuh,O=Wazuh,L=California,C=US"
plugins.security.restapi.roles_enabled:
- "all_access"
- "security_rest_api_access"

plugins.security.system_indices.enabled: true
plugins.security.system_indices.indices: [".plugins-ml-model", ".plugins-ml-task", ".opendistro-alerting-config", ".opendistro-alerting-alert*", ".opendist>

### Option to allow Filebeat-oss 7.10.2 to work ###
compatibility.override_main_response_version: true
```

Screenshot opensearch.yml

De overige opties kunnen op hun standaardinstellingen blijven staan, aangezien ze voor deze stageopdracht niet relevant zijn om aan te passen. Hieronder geef ik een korte toelichting op waarvoor de verschillende opties dienen:

- ‘cluster.name’: De naam van de cluster waarin deze node actief is.
- ‘node.max_local_storage_nodes’: Het maximaal aantal nodes op één fysieke host.
- ‘path.data’: De locatie waar Opensearch indexdata opslaat.
- ‘path.logs’: De locatie waar logbestanden opgeslagen worden.
- ‘plugins.security.ssl.http.*’: De locatie van de certificaten voor extern HTTPS-verkeer.
- ‘plugins.security.ssl.transport.*’: De locatie van de certificaten voor intern HTTPS-verkeer via API.
- ‘plugins.security.ssl.http.enabled’: Activeert HTTPS-verkeer op de API-interface.
- ‘plugins.security.ssl.transport.enforce_hostname_verification’: Bepaalt of de node verbindingen toelaat ongeacht of de hostnaam overeen komt met de naam in het certificaat.
- ‘plugins.security.ssl.transport.resolve_hostname’: Bepaalt of Opensearch het IP-adres probeert te vertalen naar een hostnaam.
- ‘plugins.security.authcz.admin_dn’: De Distinguished Name van de admin-gebruiker.
- ‘plugins.security.check_snapshot_restore_write_privileges’: Controleert of een gebruiker schrijfrechten heeft voor snapshots en herstellingen.
- ‘plugins.security.enable_snapshot_restore_privilege’: Activeert snapshot- en herstelrechten voor gebruikers.

- ‘plugins.security.nodes_dn’: De Distinguished Names van de nodes in een cluster.
- ‘plugins.security.restapi.roles_enabled’: De rollen die toegang hebben tot de REST API
- ‘plugins.security.system_indices.enabled’: Het systeem herkent en beschermt interne indexen.
- ‘plugins.security.system_indices.indices’: De indexen die het systeem herkent en beschermt

Nu de Wazuh Indexer geïnstalleerd is en de Opensearch-database correct geconfigureerd werd, kunnen we de certificaten uitrollen. Hiervoor maken we een nieuwe map aan, in dit geval ‘/etc/wazuh-indexer/certs’. Deze locatie moet overeenkomen met het pad dat in ‘opensearch.yml’ opgegeven werd. De certificaten worden vervolgens uitgepakt uit het eerder aangemaakte ZIP-bestand en in deze map geplaatst.
Hierna kan de service voor de Wazuh Indexer aangezet en gestart worden.

Ten slotte rest ons enkel nog het initialiseren en testen van de cluster. Wazuh voorziet hiervoor een script dat zich bevindt op ‘/usr/share/wazuh-indexer/bin/indexer-security-init.sh’ en dat gebruikt wordt om de cluster op te starten.
Dit script is voornamelijk bedoeld om de beveiligingsinstellingen van de Wazuh Indexer te initialiseren of bij te werken. Het focust daarbij op TLS-certificaten en gebruikersrollen, en controleert of de configuratie in ‘opensearch.yml’ correct is ingesteld.
Zo worden de certificaatgegevens ingeladen die in het configuratiebestand zijn opgegeven. Zoals eerder vermeld bevat ‘opensearch.yml’ paden naar de TLS-certificaten voor zowel HTTP- als transportbeveiliging. Daarnaast maakt het script gebruik van de rollen die eveneens in ‘opensearch.yml’ zijn gespecificeerd. Na controle initialiseert of herconfigureert het de gebruikers en rollen die nodig zijn voor een veilige communicatie tussen nodes of tussen Wazuh en de Indexer.
Tot slot zorgt het script ervoor dat een single-node of multi-node cluster op een veilige manier kan worden opgestart, met correcte authenticatie en versleuteling.
Hierna kunnen we de cluster installatie testen met het volgende commando:
curl -k -u admin:admin https://<WAZUH_INDEXER_IP_ADDRESS>:9200

Dit geeft de volgende output in de CLI:


```
{
  "name" : "node-1",
  "cluster_name" : "wazuh-cluster",
  "cluster_uuid" : "C9Pqes31QpiTe989N9MHjg",
  "version" : {
    "number" : "7.10.2",
    "build_type" : "deb",
    "build_hash" : "dae2bfc93896178873b43cdf4781f183c72b238f",
    "build_date" : "2025-04-30T10:51:28.815931460Z",
    "build_snapshot" : false,
    "lucene_version" : "9.12.1",
    "minimum_wire_compatibility_version" : "7.10.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "The OpenSearch Project: https://opensearch.org/"
}
```

Output cluster installatie

Wazuh Server

a) Installatie

De Wazuh Server bestaat uit twee grote componenten: de Wazuh Manager en Filebeat.
De Wazuh Manager is het hart van de Wazuh-opstelling. Het is verantwoordelijk voor het ontvangen, analyseren en indelen van data dat binnenkomt via de Wazuh Agents. De manager voert loganalyse uit, detecteert bedreigingen, controleert de integriteit van bestanden, voert proactieve dreigingsdetectie uit en genereert alerts. Verder beheert het ook de configuratie van Agents en de regels die aangeven of iets verdacht is.
Filebeat is een logverzamelaar die ontworpen is om logbestanden te verzenden. In Wazuh wordt Filebeat gebruikt om deze logbestanden van de Wazuh Manager naar de Wazuh Indexer door te sturen. Dit maakt het mogelijk om de gegevens te indexeren en visueel te analyseren op het Wazuh Dashboard.

Vooraleer we deze componenten kunnen installeren, moeten we opnieuw een GPG-sleutel installeren om te controleren of de softwarepakketten van Wazuh komen. Hierna kunnen we de Wazuh Manager en Filebeat installeren. Hiermee is de installatie van de Wazuh Server voltooid en kunnen we beginnen aan de configuratie.

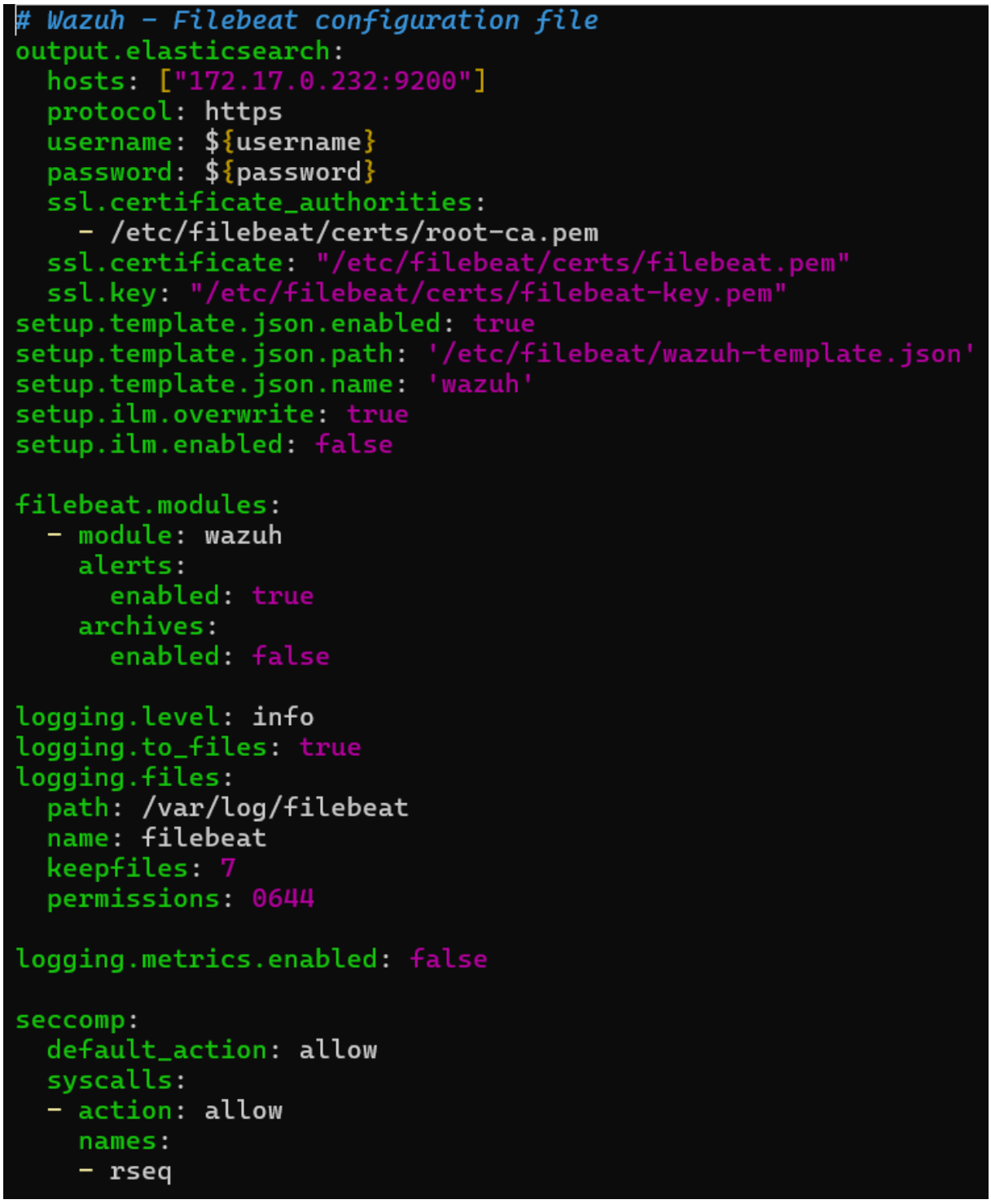
b) Configuratie

De installatiegids van Wazuh begint met het configureren van Filebeat. Hiervoor moet eerst het configuratiebestand ‘filebeat.yml’ geïnstalleerd worden. Wazuh voorziet hier een download voor die met curl geïnstalleerd kan worden. Het configuratiebestand zoals het in mijn SOC geconfigureerd is, kunt u in onderstaande schermafbeelding vinden.

Nu kan de configuratie van Filebeat beginnen. Voor mijn SOC moet enkel ‘hosts’ aangepast worden. Dit is een lijst met alle IP-adressen van de Indexers. Standaard staat dit op ‘localhost’, maar omdat onze Indexer zich op een andere VM bevindt, geven we het IP-adres hiervan in. In een multi-node cluster moeten alle IP-adressen opgegeven van de Indexers.

De overige opties kunnen op hun standaardinstellingen blijven staan, aangezien ze voor de stageopdracht niet relevant zijn om aan te passen. Hieronder geef ik een korte toelichting op waarvoor de verschillende opties dienen:

- ‘protocol’: Het protocol dat gebruikt word voor onderlinge verbinding op te zetten.
- ‘username’: De gebruikersnaam van Opensearch, wordt als variabele in de keystore meegegeven.
- ‘password’: Het wachtwoord van Opensearch, wordt als variabele in de keystore meegegeven.
- ‘ssl.certificate_authorities’: De locatie van het certificaat dat gebruikt wordt om de Opensearch certificaten te valideren.
- ‘ssl.certficate’: Cliëntcertificaat dat Filebeat gebruikt om zich als vertrouwde cliënt te identificeren.
- ‘ssl.key’: Cliëntsleutel dat Filebeat gebruikt om te bewijzen dat het de eigenaar is van het certificaat.
- ‘setup.template.json.enabled’: Schakelt een aangepast JSON-template in voor indexen.
- ‘setup.template.json.path’: Locatie waar de aangepaste JSON-template zich bevindt.
- ‘setup.template.json.name’: Naam van de aangepaste JSON-template in Opensearch.
- ‘setup.ilm.enabled’: Schakelt Index Lifecycle Management in, wat het automatisch beheer van Indexen- en logretentie is.
- ‘filebeat.modules’: Activeert een Filebeat-module.
- ‘module’: Naam van de Filebeat-module.
- ‘alerts’: Schakelt het verzamelen van Wazuh-alerts in.
- ‘archives’: Schakelt het verzamelen van gearchiveerde Wazuh-logs in.
- ‘logging.level’: Stelt het logniveau van logs in.
- ‘logging.to_files’: Schakelt in dat logoutput naar bestanden wordt geschreven
- ‘logging.files’: Logs worden opgeslagen in ‘/var/log/filebeat’ met bestandsnaam ‘filebeat’. Het bewaart tot max 7 logbestanden die elk leesrechten hebben voor iedereen en schrijfrechten voor de eigenaar.
- ‘logging.metrics.enabled’: Schakelt het loggen van prestatiestatestieken in.
- ‘seccomp’: Staat alle systeemaanroepen toe of blokkeert ze.
- ‘Syscalls’: Staat per naam systeemaanroepen toe of blokkeert ze.



```
# Wazuh - Filebeat configuration file
output.elasticsearch:
  hosts: ["172.17.0.232:9200"]
  protocol: https
  username: ${username}
  password: ${password}
  ssl.certificate_authorities:
    - /etc/filebeat/certs/root-ca.pem
  ssl.certificate: "/etc/filebeat/certs/filebeat.pem"
  ssl.key: "/etc/filebeat/certs/filebeat-key.pem"
setup.template.json.enabled: true
setup.template.json.path: '/etc/filebeat/wazuh-template.json'
setup.template.json.name: 'wazuh'
setup.ilm.overwrite: true
setup.ilm.enabled: false

filebeat.modules:
  - module: wazuh
    alerts:
      enabled: true
    archives:
      enabled: false

logging.level: info
logging.to_files: true
logging.files:
  path: /var/log/filebeat
  name: filebeat
  keepfiles: 7
  permissions: 0644

logging.metrics.enabled: false

seccomp:
  default_action: allow
  syscalls:
  - action: allow
    names:
    - rseq
```

Schermafbeelding filebeat.yml uit mijn SOC

Nu het configuratiebestand van Filebeat in orde is, moet er een keystore aangemaakt worden. Dit is een veilige opslagplaats voor gevoelige gegevens zoals wachtwoorden, API-keys en certificaten. Filebeat gaat dit gebruiken om de gebruikersnaam en het wachtwoord van Opensearch op te slaan, zodat dit niet in het configuratiebestand staat. De Filebeat keystore is onderdeel van Elastic Beats en is bedoeld voor gebruik met Elasticsearch. Omdat Opensearch een fork is van Elasticsearch, is dit grotendeels compatibel met Elastic Beats. Daardoor kunnen we hier toch gebruik van maken in het SOC.
Om de gegevens op te kunnen slaan, wordt er gebruik gemaakt van de keystore-tool. Deze tool dient om op een gebruiksvriendelijke manier aanpassingen in de keystore te maken. Als de gegevens hierin zijn opgeslagen, zijn ze bereikbaar voor alle configuratiebestanden op de Wazuh Server. Als dus ooit het wachtwoord van OpenSearch aangepast wordt, moet het enkel in de keystore veranderd worden en niet in alle aparte configuratiebestanden. Dit zorgt voor veel flexibiliteit voor de organisatie.

Om de configuratie van Filebeat af te ronden, moeten er nog enkele belangrijke bestanden gedownload worden van Wazuh. Dit zijn de Wazuh alerts template en de Wazuh module, die daarnet besproken zijn in het configuratiebestand van Filebeat.
De Wazuh alerts template is een bestand dat de algemene structuur bepaald van alerts. Het bevat informatie over hoe logdata en alerts correct gestructureerd, verwerkt en geïndexeerd moeten worden, zodat deze gegevens vlot doorzocht en gevisualiseerd kunnen worden. Het bestand krijgt ook leesrechten voor de groep van het bestand en alle andere gebruikers.
De Wazuh module is een kant-en-klare configuratie waarmee Filebeat automatisch Wazuh logs en alerts kan inlezen, verwerken en doorsturen naar Opensearch.

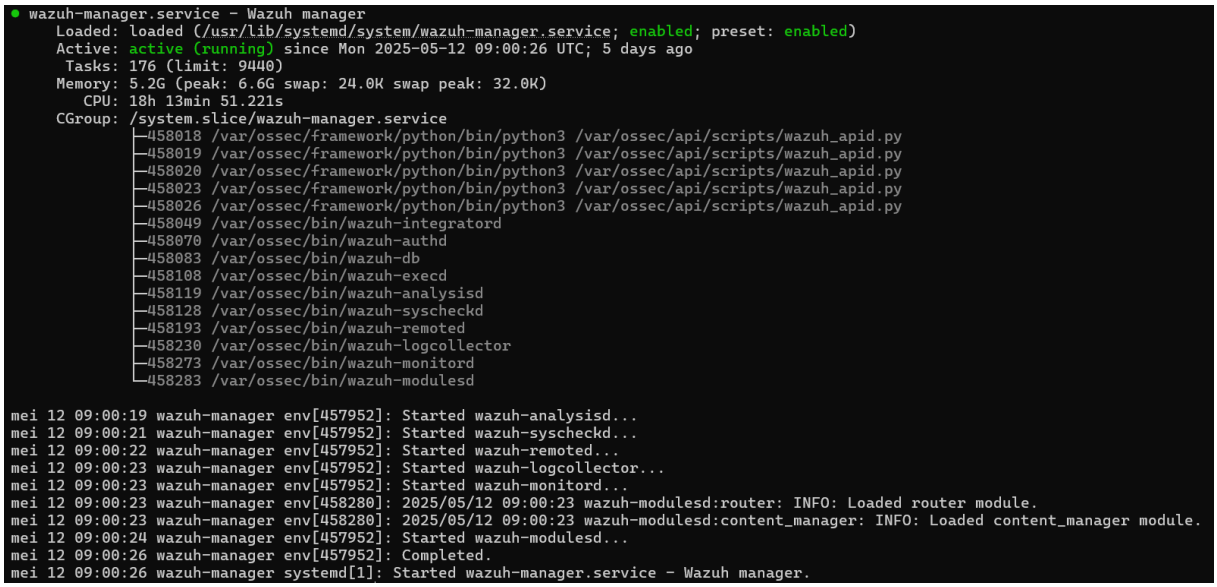
Nu Filebeat volledig geconfigureerd is, kunnen de certificaten uitgerold worden. Dit zijn dezelfde die bij de Wazuh Indexer gebruikt zijn en waarvan het TAR-bestand gekopieerd is naar deze node. De certificaten dienen in de Wazuh Server voor de verbinding tussen Filebeat en Opensearch te beveiligen, dat zojuist besproken is, en de verbinding tussen de Wazuh Manager en Wazuh Indexer, wat zo dadelijk besproken wordt.
Het proces van het uitrollen van certificaten verloopt overal hetzelfde. Het begint met het opzetten van een nieuwe map waar de certificaten in komen. In de Wazuh Server wordt dit bijgehouden in '/etc/filebeat/certs' dat lees- en uitvoerrechten krijgt voor de eigenaar van de map. Het TAR-bestand wordt uitgepakt naar deze locatie en de eigenaar van de certificaten krijgt hier leesrechten op. De eigenaar van zowel de map en de bestanden hierin wordt aangepast naar het rootaccount.

De laatste grote configuratiestap die nog resteert, is het configureren van de Wazuh Manager. Dit is nodig om een connectie op te zetten tussen de Wazuh Server en Wazuh Indexer. De Wazuh Server bezit namelijk het centrale configuratiebestand 'ossec.conf' wat bepaalt hoe de Wazuh Server en Agent zich gedragen en hoe deze met andere componenten communiceren.
Vooraleer we hier aanpassingen aan gaan maken, moeten eerst gegevens worden toegevoegd aan de Wazuh keystore. Deze keystore hanteert hetzelfde principe als de Filebeat keystore, maar dan voor de Wazuh Manager. De inloggegevens van de Wazuh Indexer worden toegevoegd aan deze keystore met de wazuh-keystore tool. Deze inloggegevens gaan later gebruikt worden in configuratiebestanden om een connectie te maken met de Wazuh Indexer.
Hierna kan de configuratie in 'ossec.conf' aangepast worden. Hierin wordt het blok 'Indexer' aangepast, dat u hieronder kunt zien, met de nieuwe informatie. In het blok 'hosts' wordt het IP-adres van de host aangepast naar het IP-adres van de Wazuh Indexer. Bij een multi-node cluster kunnen hier nog meerdere hosts en hun IP-adres aan toegevoegd worden.
De blok 'ssl' bevat de paden naar de locaties van de certificaten om de communicatie tussen de Wazuh Manager en Indexer veilig te maken. Dit is voor het SOC niet aan aangepast en blijft op de standaardinstellingen staan.
Aangezien het volledige 'ossec.conf'-bestand meerdere pagina's in beslag neemt, ga ik hier niet alles van toelichten. U kunt het volledige bestand vinden in Bijlage 1.

```
<indexer>
  <enabled>yes</enabled>
  <hosts>
    <host>https://172.17.0.232:9200</host>
  </hosts>
  <ssl>
    <certificate_authorities>
      <ca>/etc/filebeat/certs/root-ca.pem</ca>
    </certificate_authorities>
    <certificate>/etc/filebeat/certs/filebeat.pem</certificate>
    <key>/etc/filebeat/certs/filebeat-key.pem</key>
  </ssl>
</indexer>
```

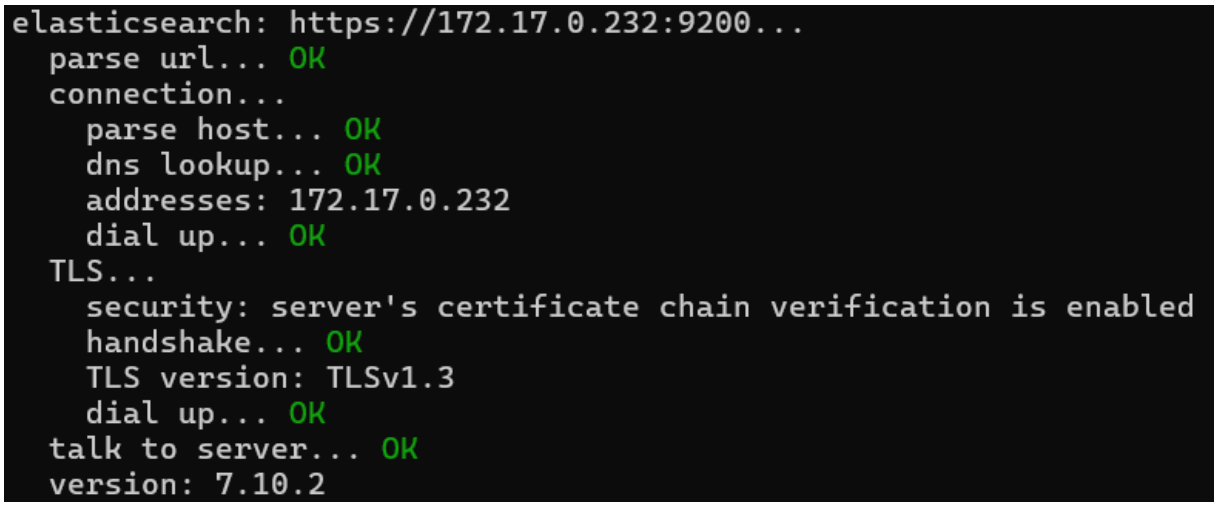
Schermafbeelding uit ossec.conf: Indexer

Alle configuraties zijn nu gebeurt en rest enkel nog om alle services aan te zetten en te testen of het werkt. Eerst wordt de wazuh-manager service aangezet. Deze controleert of de configuratie goed ingesteld is en voert dit uit. Bij aanpassingen in de toekomst aan de configuratie, zal deze service ook telkens opnieuw opgestart moeten worden. Het wordt aangeraden om te testen of de service goed is opgestart om verdere problemen te voorkomen. Als er toch fouten zijn, worden deze ook weergeven in de logs van de wazuh-manager service.
Hieronder vindt u een schermafbeelding van de wazuh-manager service als hij succesvol aanstaat.



Schermafbeelding status wazuh-manager service

Als de wazuh-manager goed is opgestart, kan de service van filebeat aangezet worden. Net als bij de wazuh-manager service controleert de filebeat service of de configuratie goed ingesteld is en voert dit uit. Om te controleren of de service goed is opgestart, heeft Filebeat zijn eigen test. Dit is ‘filebeat test output’ dat de verbinding met Opensearch en de TLS-certificaten gaat testen.
Hieronder kunt u in de schermafbeelding de output van dit commando zien als alles goed verloopt.

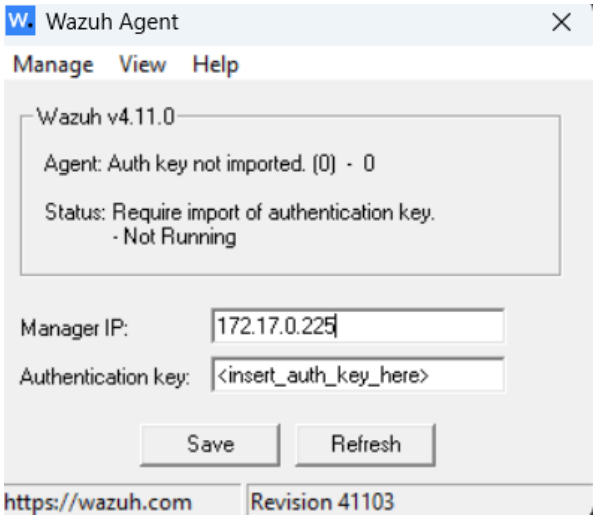


Wazuh Agent

De Wazuh Manager is natuurlijk niks zonder een Wazuh Agent waar de logs en alerts vandaan komen. Omdat de Wazuh Agent op verschillende besturingssystemen geïnstalleerd kan worden, voorziet Wazuh hiervoor verschillende installatiegidsen. Op mijn stageplaats wordt vooral gebruik gemaakt van Windows en een paar Linux-servers. Daarom worden enkel de installaties hiervan besproken in dit realisatiedocument.

a) Windows

De installatie op Windows gebeurt met een EXE-bestand dat geïnstalleerd kan worden vanuit de Wazuh website. Dit bestand installeert de Wazuh Agent op het endpoint. Dit geeft een configuratiescherm, dat u in de afbeelding hieronder ziet, waarin het IP-adres van de Wazuh Server ingevuld moet worden.



Schermafbeelding configuratiescherm Wazuh Agent

Als de gebruiker dit scherm niet opent, moet het handmatig toegevoegd worden aan het 'ossec.conf' bestand. Dit bestand bevindt zich in 'C:\Program Files (x86)\ossec-agent'. Hier kunnen echter wel enkele problemen opkomen met toegang tot dit bestand voor gebruikers. Daarom heb ik een uitgebreide handleiding geschreven voor het MPI Oosterlo. Deze kunt u vinden in Bijlage x.

Ten slotte moet de wazuh-agent service opnieuw opgestart worden om de installatie te voltooien. Na enkele minuten is het nieuwe endpoint toegevoegd.

b) Linux

De installatie op Linux is korter met minder stappen dan de installatie op Windows. Het begint met het installeren van

Wazuh Dashboard

c) Installatie

Het laatste component dat nog geïnstalleerd moet worden, is het Wazuh Dashboard. Dit is het component dat centraal alles toont dat op de andere componenten gedaan wordt. De logs en alerts die door Wazuh Agent naar de Wazuh Server gestuurd worden, kunnen hier bekeken worden en gefilterd op de indexen bepaalt door de Wazuh Indexer.

De installatie van het Wazuh Dashboard begint, net als bij alle andere installaties, met het installeren van de GPG-sleutel. Nu kan de installatie gevalideerd worden en kan het Wazuh Dashboard geïnstalleerd worden. Wazuh voorziet hiervoor een pakket waarin alle bestanden zitten, zodat direct na de installatie aan de configuratie begonnen kan worden.

d) Configuratie

Om het Wazuh Dashboard te configureren, voorziet Wazuh het configuratiebestand 'opensearch_dashboards.yml' dat zich in '/etc/wazuh-dashboard' bevindt.

In het configuratiebestand wordt als eerste 'server.host' aangepast. Hier komt het IP-adres van het Wazuh Dashboard te staan waarmee de andere hosts en gebruikers kunnen verbinden.
Het tweede dat wordt aangepast is 'opensearch.hosts'. Dit zijn de hosts waar een Wazuh Indexer op draait. Het Wazuh Dashboard moet hiermee kunnen verbinden om data en de indexering hiervan op te halen.

De overige opties kunnen op hun standaardinstellingen blijven staan, aangezien ze voor de stageopdracht niet relevant zijn om aan te passen. Hieronder geef ik een korte toelichting op waarvoor de verschillende opties dienen:

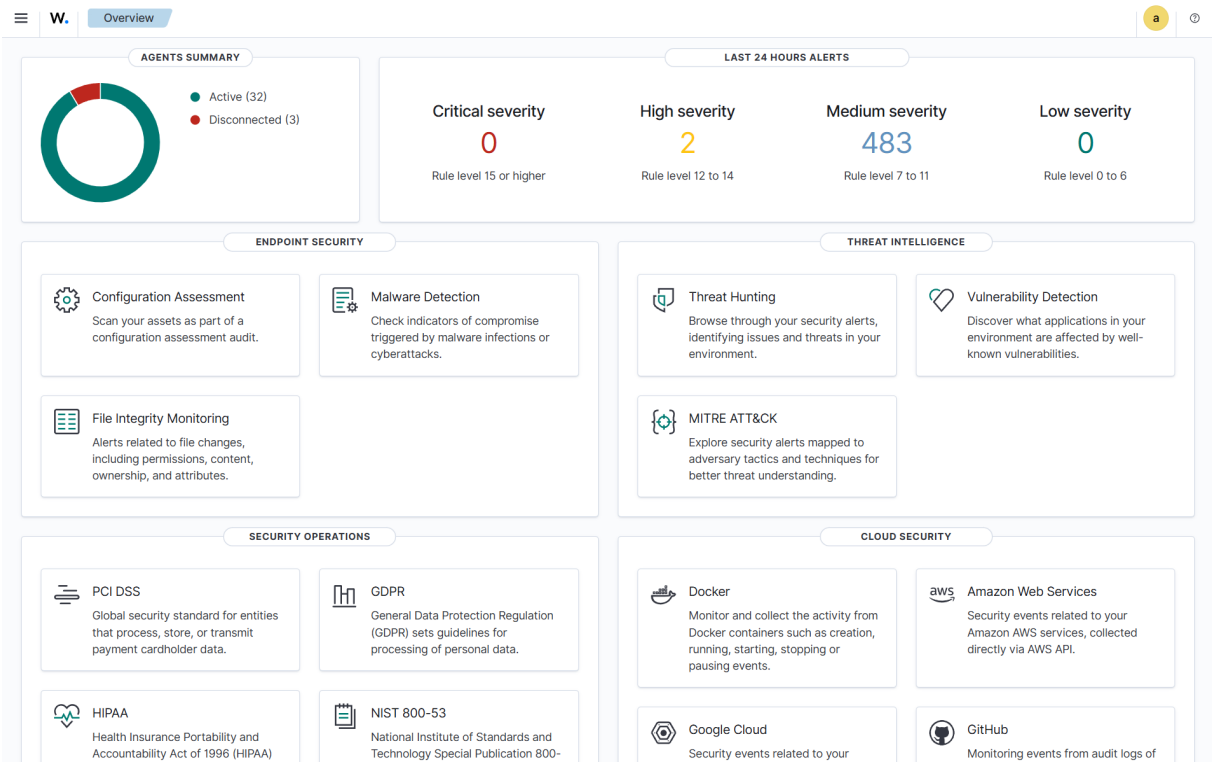
- 'server.port': De poort waarop het Dashboard draait.
- 'opensearch.ssl.verificationMode': De manier waarop er gecontroleerd wordt op SSL (TLS)
- 'opensearch.requestHeadersAllowlist': De lijst HTTP-headers die toegestaan zijn om van het Dashboard naar Opensearch door te sturen.
- 'opensearch_security.multitenancy.enabled': Bepaalt of er meerdere data-omgevingen voor meerdere gebruikers mag zijn.
- 'opensearch_security.readonly_mode.roles': Rollen voor gebruikers die enkel het Dashboard mogen lezen.
- 'server.ssl.enabled': Schakelt SSL in voor HTTPS verbindingen.
- 'server.ssl.key': De locatie van de private sleutel.
- 'server.ssl.certificate': De locatie van het certificaat.
- 'opensearch.ssl.certificateAuthorities': De locatie van het CA-certificaat dat het Opensearch SSL-certificaat controleert.
- 'uiSettings.overrides.defaultRoute': De standaardpagina die geladen wordt bij het openen van het Dashboard

```
server.host: 172.17.0.233
server.port: 443
opensearch.hosts: https://172.17.0.232:9200
opensearch.ssl.verificationMode: certificate
#opensearch.username:
#opensearch.password:
opensearch.requestHeadersAllowlist: ["securitytenant","Authorization"]
opensearch_security.multitenancy.enabled: false
opensearch_security.readonly_mode.roles: ["kibana_read_only"]
server.ssl.enabled: true
server.ssl.key: "/etc/wazuh-dashboard/certs/dashboard-key.pem"
server.ssl.certificate: "/etc/wazuh-dashboard/certs/dashboard.pem"
opensearch.ssl.certificateAuthorities: ["/etc/wazuh-dashboard/certs/root-ca.pem"]
uiSettings.overrides.defaultRoute: /app/wz-home
```

Schermafbeelding opensearch_dashboards.yml uit mijn SOC

DASHBOARD

Op onderstaande afbeelding kunt u het dashboard zien na de installatie van alle componenten. Het bevat veel informatie dat in aparte vakken wordt geplaatst om een duidelijk overzicht te bieden aan de gebruiker. In dit deel van mijn document ga ik de belangrijkste vakken overlopen en wat deze doen.



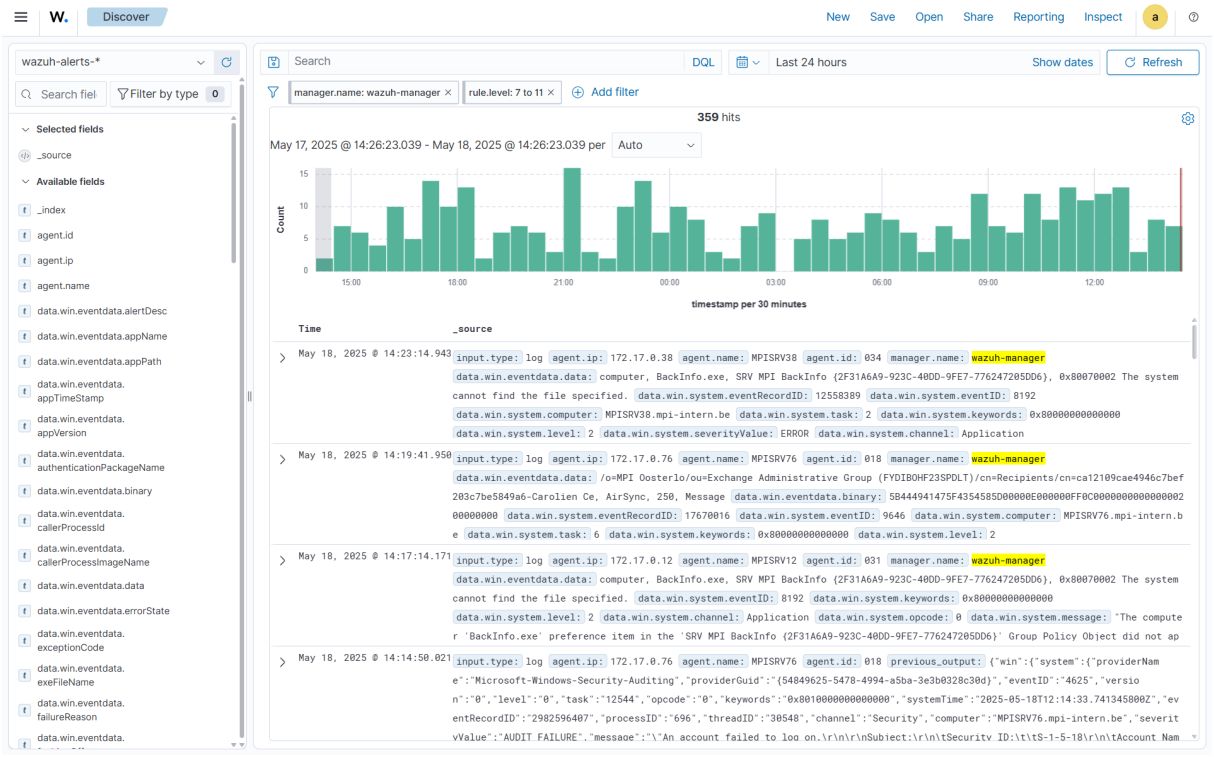
Wazuh Dashboard van MPI Oosterlo

Alerts

Het belangrijkste tabblad dat als tweede zichtbaar is op het Dashboard zijn de Alerts. Hier staan alle alerts en logs die Wazuh heeft binnengekregen binnen een bepaalde tijdsduur (standaard 24 uur). Op de schermafbeelding kunt u alle alerts zien die in de afgelopen 24 uur op mijn SOC gegenereerd zijn en die als niveau ‘medium severity’ hebben gekregen. Wazuh heeft vier categorieën op basis van meldingsniveau:

- Low severity: niveau van 0-6
- Medium severity: niveau van 7-11
- High severity: niveau van 12-14
- Critical severity: niveau van 15

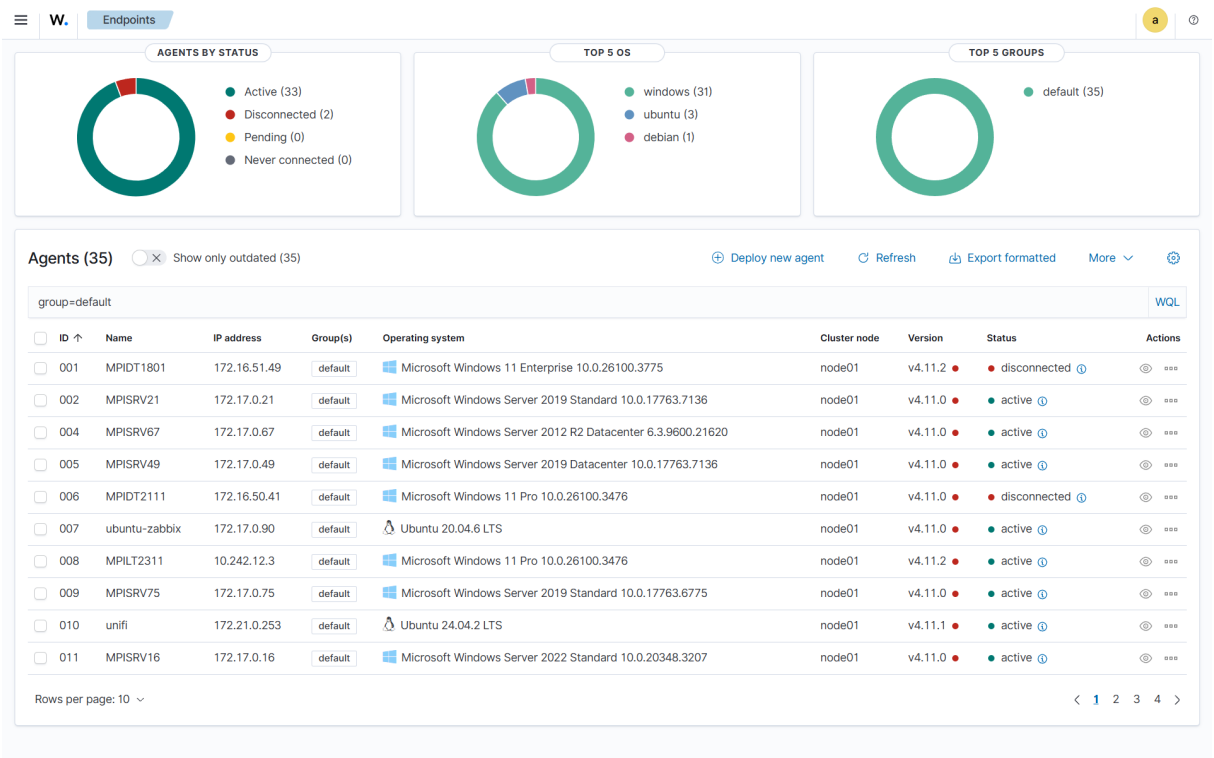
Het hoofdonderdeel van dit tabblad is natuurlijk om alle alerts te laten zien. Het doet dit door ze onder elkaar te plaatsen en met telkens een kleine uitleg waarover de alert gaat. Deze alerts kunnen opengedaan worden om meer informatie te tonen. Deze informatie wordt getoond in de vorm van een tabel of in JSON-formaat. Aan de hand hiervan kan er bepaald worden hoe de alerts opgelost kunnen worden.
Als er dan een oplossing gevonden is, is het makkelijk om te weten of dit probleem zich elders ook voordoet. Doordat de Wazuh Indexer alle data al geïndexeerd heeft, kan de gebruiker in het Dashboard makkelijk filteren om deze indexen. Dit kan de naam van de agent zijn of het type alert dat gegenereerd is.
Een andere manier waarop de alerts getoond worden, is in de grafiek die boven de alerts staat. Deze geeft de alerts weer die per half uur gegenereerd zijn. Als er dan momenten zijn met veel alerts of iemand wil kijken of de alert zich vaker voordoet in korte tijdsspanne, kan er gefilterd worden door op de balk van de grafiek te klikken. Wazuh toont dan enkel de alerts die zich in die periode hebben voorgedaan.
Wazuh kan ook niet enkel de afgelopen 24 uur laten zien. Er kan gefilterd worden tot meerdere dagen en ook kleinere intervallen zijn mogelijk. Dit voorziet flexibiliteit voor als een gebruiker wil kijken wat er tijdens het weekend gebeurd is.



Schermafbeelding tabblad Alerts

Agent Summary

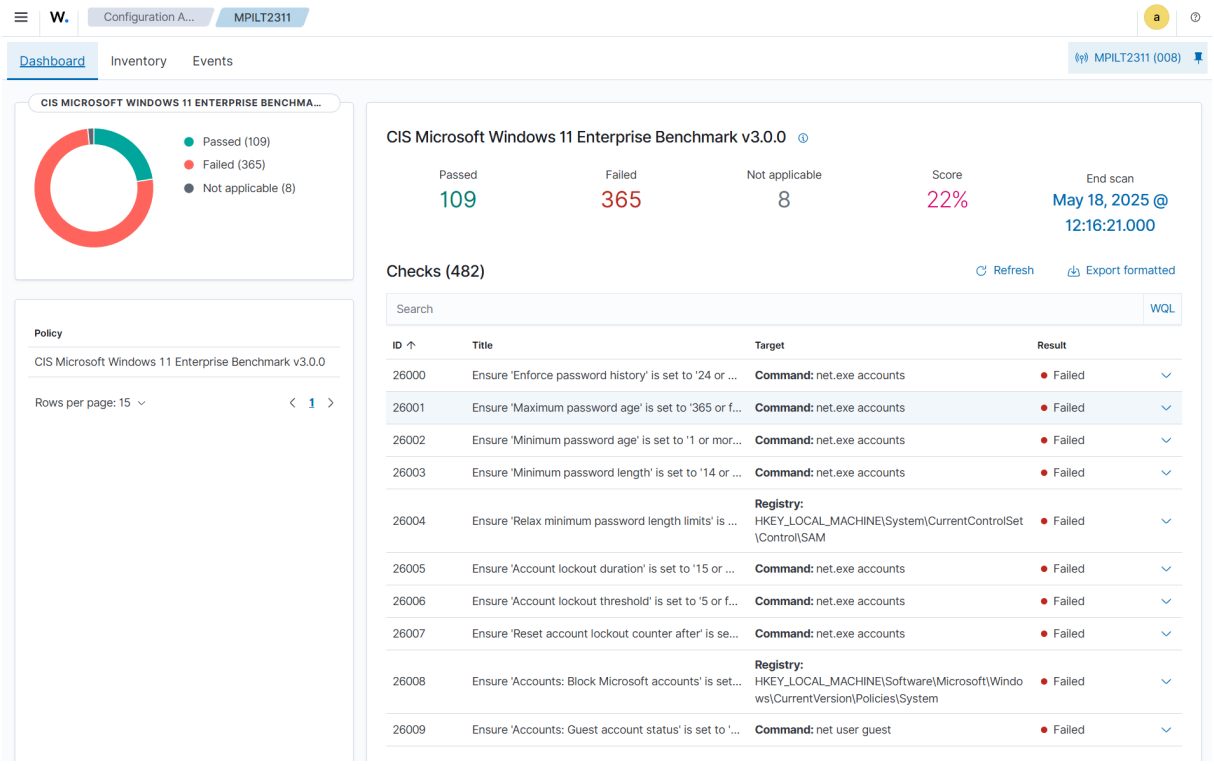
Een van de belangrijkste onderdelen van Wazuh en die opvalt als een gebruiker het Dashboard opent is de Agent Summary. Hierin krijgt de gebruiker een overzicht te zien van alle endpoints die verbonden zijn met de Wazuh Server. Van deze endpoints krijgt Wazuh alerts en logs doorgestuurd voor analyse. Op de schermafbeelding hieronder kunt u zien hoe dit onderdeel eruitziet.
Om te beginnen zien we linksboven de status van alle endpoints. Meestal is dit Active of Disconnected, maar bij het toevoegen van een nieuwe endpoint komt deze eerst in Never connected en daarna in Pending. Een endpoint die status Disconnected heeft, betekent meestal dat de endpoint uitstaat. Er kan dus hier gecontroleerd worden of bepaalde servers uitvallen.
Naast de status van de endpoints staan top vijf meest gebruikte besturingssystemen. Dit is handig om te weten welke besturingssystemen het vaakst voorkomen. Als er dan een nieuwe CVE ontdekt wordt, kan er hier snel gecontroleerd worden hoeveel en welke servers bijgewerkt moeten worden.
Agents kunnen ingedeeld worden in verschillende groepen. Als een omgeving veel servers hebben met dezelfde functie, kunnen die in een groep gezet worden. Zo kan er bijvoorbeeld makkelijk naar alle fileservers gezocht worden. Aangezien het MPI Oosterlo geen grote omgeving heeft, was de onderverdeling in groepen niet van toepassing.
Het laatste tabblad toont alle Agents. Hier kan per Agent informatie gevonden worden over bijvoorbeeld het besturingssysteem, de naam, het IP-adres en de status. Het biedt een makkelijk overzicht over je omgeving en ze kunnen per 10, 25, 50 of 100 Agents getoond worden. Via hier kan er ook een Agent geopend worden. Dit toont dan de resultaten van de andere blokken, maar dan voor de specifieke Agent. Ook kan hier de Inventory Data gevonden worden die bijvoorbeeld de open poorten weergeeft.
Ten slotte is het ook mogelijk om via hier een nieuwe Agent toe te voegen. Hier moeten er dan enkele gegevens ingevuld worden zoals het besturingssysteem en het IP-adres. Aan de hand van deze informatie genereerd Wazuh een commando dat na het uitvoeren de Wazuh Agent toevoegt.



Schermafbeelding tabblad Agent Summary

Configuration Assessement

Om te testen hoe veilig hoe een Agent is, doet Wazuh testen om een veiligheidsscore te bepalen. Deze veiligheidsscores zijn gebaseerd op CIS Benchmarks die met behulp van een script gaan testen of een endpoint aan hun vereisten voldoet. Zoals u op de schermafbeelding hieronder kunt zien, liggen deze scores vaak niet hoog. Dit komt omdat op persoonlijke toestellen, zoals de laptop die u hieronder kunt zien, er genoeg vrijheid moet zijn voor de gebruiker om niet te hard gehinderd te worden. Als de gebruiker bij elke actie hun wachtwoord moet ingeven, dan is het systeem wel veilig maar niet bruikbaar.
Toch is het handig om deze benchmark in de gate te houden en er bepaalde zaken van te implementeren. Op kritieke servers waar weinig veranderingen op gebeuren is het beter om een hogere score te hebben. De vereiste om gebruiksvriendelijk te zijn ligt hier lager dan de vereiste om veilig te zijn.
Wazuh laat zien welke testen het heeft uitgevoerd en welke hiervan geslaagd en gefaald zijn. Een gebruiker kan makkelijk kijken welke maatregelen het kan implementeren om hun systeem beter te beveiligen.



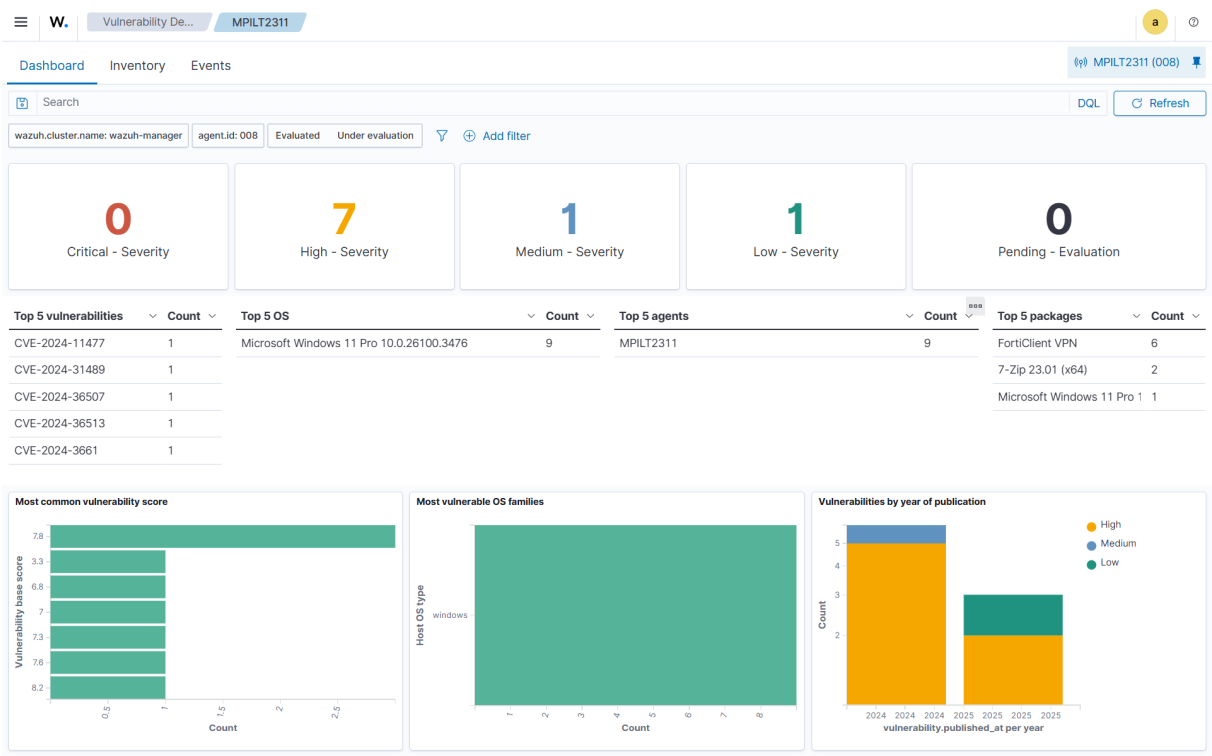
Schermafbeelding tabblad Configuration Assessement

Vulnerability Detection

CVE's zijn kwetsbaarheden die gevonden zijn in besturingssystemen en software. Ze worden geclassificeerd door de MITTRE ATT&CK framework zodat alle kwetsbaarheden dezelfde classificering hanteren. Deze CVE's zijn belangrijk om aanvallen tegen te gaan en systemen veilig te houden. Daarom biedt Wazuh het tabblad 'Vulnerability Detection' aan waar de CVE's per Agent zichtbaar zijn.
Wazuh deelt de CVE's in vijf categorieën op basis van CVSS-score:

- Critical: Score van 9.0-10.0
- High: Score van 7.0-8.9
- Medium: Score van 4.0-6.9
- Low: Score van 0.1-3.9
- Pending evaluation: Hebben nog geen score ontvangen

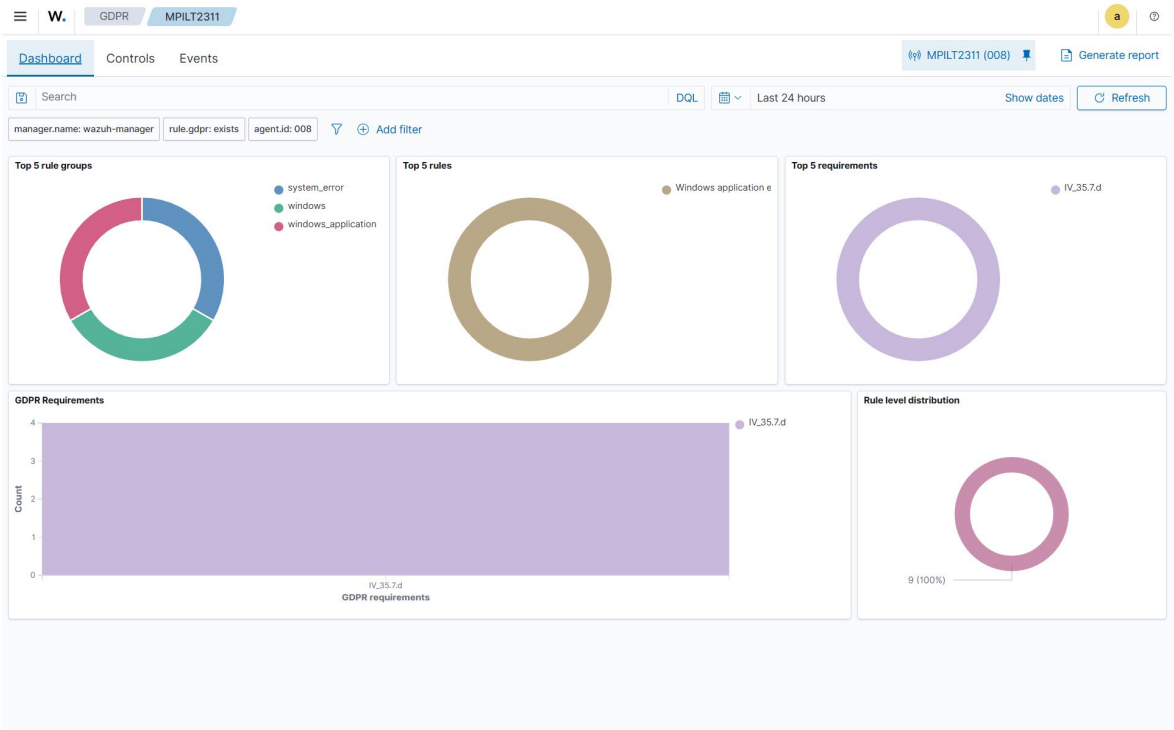
Onder deze categorieën laat Wazuh zien welke CVE's het meeste voorkomen en bij welk besturingssysteem. Het laat ook de package zien die de kwetsbaarheid bezit.
Hoewel Wazuh geen directe oplossing biedt, kan er zelf gezocht worden naar de CVE om een oplossing te vinden.



Schermafbeelding van tabblad Vulnerability Detection

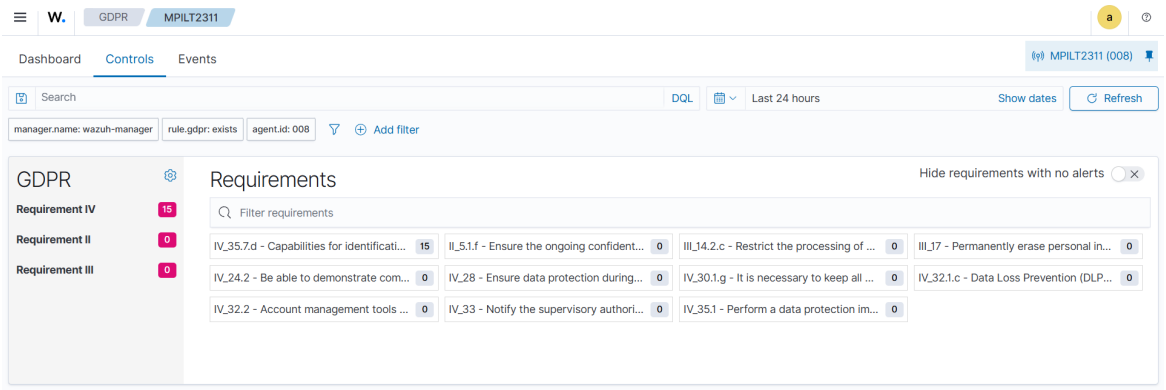
GDPR

Het laatste tabblad dat ik nog wil bespreken is de GDPR. Dit is de General Data Protection Regulation die is opgesteld door de Europese Unie voor het beschermen van data van de Europese burgers. Het MPI Oosterlo verwerkt veel cliëntengegevens, waardoor data beveiliging van cruciaal belang is. Wazuh gebruikt de GDPR om te controleren hoe veilig Agents zijn en hoe goed ze voldoet aan de wetgeving.
Om dit te controleren gebruikt Wazuh de alerts en logs die het binnenkrijgt en analyseert of het voldoet aan de wetgevingen van de GDPR. Als er ergens een fout is, laat Wazuh zien welke regel het niet aan voldoet. Op het onderstaande Dashboard laat Wazuh zien welke regelgevingen het vaakst niet aan voldoen worden en welke alerts eraan gelinkt zijn. Dit biedt een duidelijk overzicht voor de gebruikers en zo kunnen ze ook snel zien welke regelgevingen ze het minste aan voldoen. Het Dashboard kan zowel individuele Agents laten zien, zoals op de schermafbeelding hieronder, of alle Agents samen.



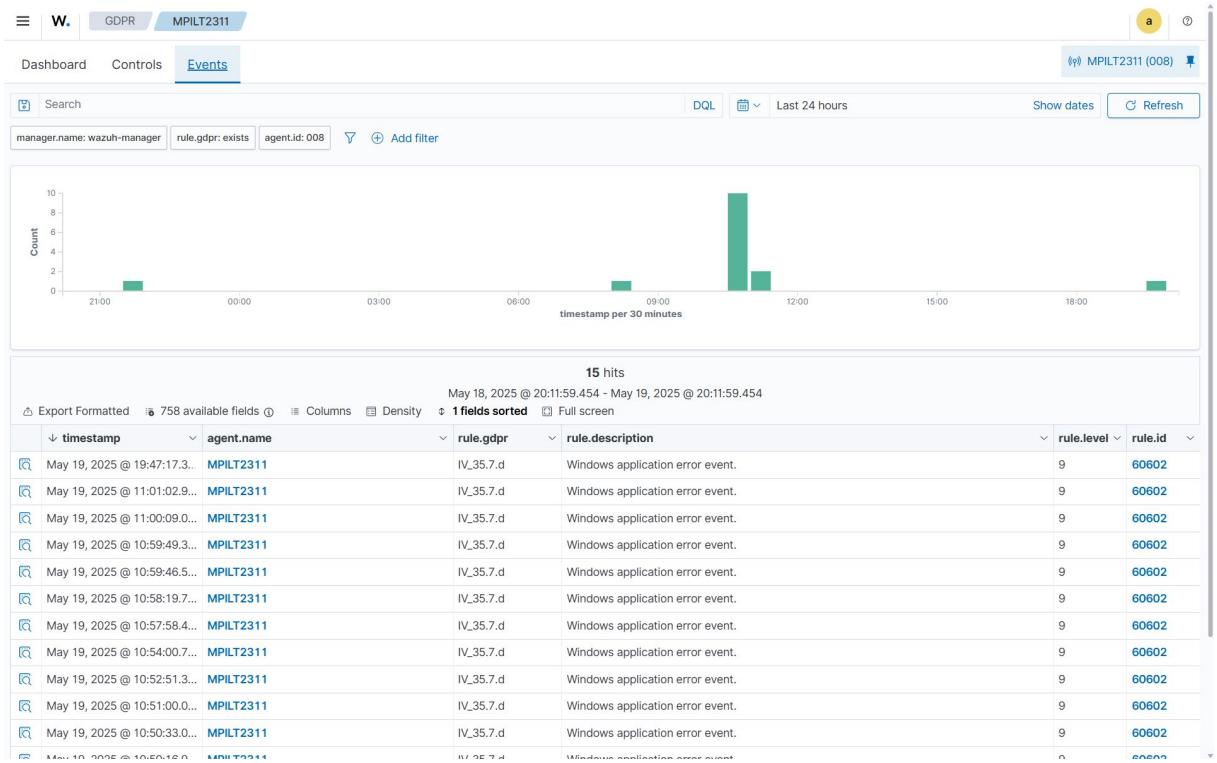
Schermafbeelding van tabblad GDPR > Dashboard

Als een gebruiker meer informatie wil over de verschillende regelgevingen, kan er op gekeken worden bij Controls. Hier staan alle vereisten waar Wazuh op controleert en hoeveel alerts hieraan gelinkt zijn. Bij het openklikken van een vereiste, staat er een beschrijving over de vereiste en welke alerts eraan gelinkt zijn. Met deze informatie kan de gebruiker opzoek gaan naar een manier om zowel de alert op te lossen alsook te voldoen aan de GPDR.



Schermafbeelding van tabblad GPDR > Controls

Ten slotte is er nog het tabblad Events. Hier krijgt de gebruiker een volledig overzicht over alle alerts en de vereiste van de GDPR die daaraan gelinkt is. Het toont dit op dezelfde manier als het alerts toont in het Alerts tabblad. Er kan hier ook makkelijk gefilterd worden op verschillende indexen zodat er een beeld gevormd kan worden hoe vaak er niet aan een vereiste voldaan wordt en of dit terugkerende problemen zijn.



Schermafbeelding tabblad GDPR > Events

Natuurlijk zijn dit niet de enige functies die Wazuh te bieden heeft. Het heeft tal van functies die nuttig zijn voor verschillende organisaties en gebruikers. Ik heb ervoor gekozen om deze functies te laten zien, omdat deze de belangrijkste zijn voor mijn SOC en waar het MPI Oosterlo het meeste mee in aanraking gaat komen.

CUSTOM RULES

Na het opzetten van Wazuh met zijn Agents en het analyseren van alerts op het Dashboard, kan het zijn dat er toch nog aanpassingen moeten gebeuren. Sommige alerts krijgen misschien niet het juiste meldingsniveau en moeten verhoogd worden. Andere alerts komen misschien te vaak voor op korte tijd en moeten beperkt worden om flooding te voorkomen. Hiervoor heeft Wazuh een oplossing: het maken van custom rules. Dit zijn rules die bepalen hoe alerts gelogd moeten worden of nieuwe alerts genereren vanuit meerdere andere alerts. Een voorbeeld dat ik zelf heb gemaakt is een hoger meldingsniveau geven als er meerdere gefaalde inlogpogingen zijn over SSH. Normaal geeft dit meldingsniveau 10, maar omdat dit dan niet verder in mijn SOC wordt opgenomen, heb ik hier 13 van gemaakt als er meerdere alerts op korte tijd gedetecteerd worden. Deze custom rules worden in '/var/ossec/etc/rules/local_rules.xml' aangemaakt.

Hieronder ziet u een schermafbeelding van het 'local_rules.xml' bestand. Alle rules bevinden zich in de groep 'custom'. Er zijn twee soorten rules opgesteld in mijn SOC:

- 'ignore rule': Deze rule houdt bij hoe vaak een ID van een alert gedetecteerd wordt. Als er te veel binnenkomen binnen de aangegeven tijd, dan wordt de alert genegeerd voor een bepaalde tijd.
- 'elevation rule': Deze rule geeft alerts die een te laag meldingsniveau krijgen, een hoger meldingsniveau.

Voor de 'ignore rule' heb ik de alert die gegenereerd werden door een ongeldige RADIUS cliënt genegeerd. Deze alerts zorgden namelijk voor overlast op Wazuh waardoor er telkens dezelfde alerts binnenkwamen. Daarom is de rule opgesteld dat als er twee meldingen gegeven werden binnen 60 seconden, dat de alert 900 seconden genegeerd zou worden. Dit zorgde ervoor dat er zeker genoeg tijd was om de nodige alerts te detecteren en de tijdelijk te dempen. De 'ignore rule' is handig omdat er geen alert verloren gaat, maar dat een enkele alert niet alle andere overschaduwd omdat er te veel gegenereerd worden.

De 'elevation rule' heb ik gebruikt om een SSH brute force aanval te detecteren op de Agents. Dit is een aanval die probeert de juiste inloggegevens te vinden door veelgebruikte gebruikersnamen en wachtwoorden te testen. Als de aanvaller erin slaagt om de juiste inloggegevens te vinden, kunnen ze met het SSH-protocol toegang krijgen tot de het systeem. Deze alert werd eerst enkel met meldingsniveau 10 gegeneerd, wat te weinig is aangezien dit onder 'medium'-alerts valt. Daarom is er een rule gemaakt die een nieuwe alert genereerd als er vier alerts met meldingsniveau 10 gedetecteerd worden binnen de 60 seconden. Zo valt de bedreiging harder op en kunnen er stappen genomen worden om dit tegen te gaan. Wazuh heeft verschillende ID's voor deze alert. Daarom heb ik voor de meest voorkomende ID's een rule toegevoegd, zodat er zeker geen gemist worden.

```
<group name="custom, ignore">
  <rule id="100002" level="10" frequency="2" timeframe="60" ignore="900">
    <if_matched_sid>61110</if_matched_sid>
    <description>The Trend Micro Unauthorized Change Prevention Service service depends on the tmactmon service which failed to start because of the follow>
The dependency service or group failed to start.</description>
  </rule>

  <rule id="100003" level="13" frequency="4" timeframe="60">
    <if_matched_sid>5551</if_matched_sid>
    <description>PAM: Multiple failed logins in a small period of time. Possible brute force attack.</description>
  </rule>

  <rule id="100004" level="13" frequency="4" timeframe="60">
    <if_matched_sid>2502</if_matched_sid>
    <description>PAM: Multiple failed logins in a small period of time. Possible brute force attack.</description>
  </rule>
  <rule id="100005" level="13" frequency="4" timeframe="60">
    <if_matched_sid>5758</if_matched_sid>
    <description>PAM: Multiple failed logins in a small period of time. Possible brute force attack.</description>
  </rule>
</group>
```

Schermafbeelding van rules uit 'local_rules.xml'

3.1.2. Graylog + Fluentbit

Een nadeel van Wazuh is dat logs geen uniform formaat hanteren. Dit betekent dat de indexen die gecreëerd worden door de Wazuh Indexer, niet altijd dezelfde namen hebben voor hetzelfde stuk data. In mijn SOC heb ik als voorbeeld het IP-adres waar een aanval vandaan komt. Wazuh geeft bij Windows de index ‘data.win.eventdata.clientIPAddress’ en bij Linux ‘data.srcip’. Zo wordt het moeilijk om hier automatisch op te filteren en mee te werken. Daarom ben ik op zoek gegaan naar een manier om de indexen aan te passen en één universele te hebben, zoals bijvoorbeeld ‘srcip’.

Wazuh zelf biedt hier geen oplossing voor aan

3.2. SOAR

3.2.1. Shuffle

SETUP

De installatie van Shuffle gebeurt typisch via Docker met Docker Compose, maar is ook mogelijk met Kubernetes. Beide opties zorgen voor een efficiënte installatie en het eenvoudig beheren van de verschillende onderdelen van het platform. Voor het SOC is de keuze gemaakt om te werken met Docker, omdat dit de meest gebruikte manier is.

Shuffle bestaat uit meerdere componenten, zoals de frontend, backend en database, die elk hun eigen Docker container hebben. Deze worden opgezet met het Docker Compose. Dit is een service die alle informatie heeft over de verschillende containers en welke vereisten ze hebben. Door dit te gebruiken kunnen de verschillende componenten afzonderlijk opgezet worden, wat het eenvoudiger maakt om ze in de toekomst afzonderlijk te beheren.

Docker Compose zorgt er ook voor dat de omgeving op een gestructureerde manier wordt opgezet. Het zorgt ervoor dat alle componenten met elkaar verbonden zijn en correct geconfigureerd zijn. Hierdoor biedt Shuffle de mogelijkheid om snel opgezet te worden en kunnen gebruikers er direct mee aan de slag gaan. Dit betekent echter niet dat er zich geen problemen kunnen voordoen, waar ik zo meteen meer over vertel.

a) Installatie

Vooraleer er aan de installatie begonnen kan worden, moeten eerst de nodige tools geïnstalleerd zijn. De installatie van Shuffle gebeurt via Docker met Docker Compose, wat betekent dat we dit moeten installeren. Dit kan gebeuren met behulp van de documentatie van Docker.
Docker heeft enkele tools nodig vooraleer het geïnstalleerd kan worden. Net als bij Wazuh heeft het ook een GPG-sleutels om de installatie te controleren. Deze worden in de map '/etc/apt/keyrings' gestoken en de map krijgt leesrechten voor alle gebruikers. Hierna kan het Docker repository toegevoegd worden aan de Apt-bronnen van Ubuntu. Hierdoor kan Docker geïnstalleerd en bijgewerkt worden met het apt-commando.
Er kan nu verder worden gegaan met het installeren van Docker. Dit bevat verschillende Docker-componenten, zoals Docker Compose en de Docker CLI. Deze componenten zorgen ervoor dat alle functionaliteiten van Docker mogelijk zijn. Dit kan getest worden door de standaardcontainer van Docker te runnen genaamd 'hello-world'. Hieronder kunt u zien wat dit weergeeft.

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:dd01f97f252193ae3210da231b1dca0cffab4aadb3566692d6730bf93f123a48
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Schermafbeelding Docker container 'hello-world'

Hierna kan de installatie van Shuffle zelf beginnen. Shuffle heeft een Github repository waar alle nodige bestanden zich in bevinden. Deze repository kopiëren we naar de VM waarop Shuffle geïnstalleerd wordt. In de repository staat een docker-compose.yml-bestand. In dit bestand staan alle configuraties van de verschillende componenten uitgeschreven. Door dit bestand uit te voeren met Docker Compose, gaan alle componenten correct opgezet worden en kan er begonnen worden met Shuffle. Onder normale omstandigheden zou er niks aangepast moeten worden aan dit document. Hieronder licht ik wel per component toe wat het doet en hoe het geconfigureerd is.

Frontend

De frontend van Shuffle is de webinterface waarmee gebruikers op een visuele manier kunnen werken met het platform. Via deze interface kunnen gebruikers workflows opbouwen, beheren en uitvoeren. De frontend vormt de visuele laag bovenop de achterliggende API's die Shuffle aansturen. Hoewel de logica en uitvoering achter de schermen plaatsvinden, maakt de frontend het mogelijk om dit alles op een gebruiksvriendelijke en overzichtelijke manier te bedienen.

Op de onderstaande schermafbeelding kunt u de configuratie van de frontend zien in de docker-compose.yml. Hieronder licht ik kort toe wat de opties doen:

- 'image': De Docker image waarop de frontend gebaseerd is, Shuffle heeft zijn eigen frontend image die hiervoor gebruikt wordt.
- 'container_name': De naam die gegeven wordt aan de Docker container voor makkelijk de container te herkennen.
- 'hostname': De hostnaam die gegeven wordt aan de Docker container voor interne communicatie.
- 'ports': De poorten die gebruikt worden voor HTTP en HTTPS. Deze kunnen zelf gekozen worden door het aan te passen in het .env-bestand, maar zijn standaard 80 en 443.
- 'networks': Het interne netwerk waarmee de frontend verbonden is en communiceert met de andere componenten.
- 'environment': Omgevingsvariabele waar de hostnaam van de backend ingegeven wordt. Zo weet de frontend hoe het met de backend kan verbinden en wordt dit toegevoegd aan het .env-bestand.
- 'restart': Docker herstart de container automatisch als het stopt, tenzij dit door de gebruiker gebeurt.
- 'depends_on': Welk component eerst geconfigureerd moet zijn vooraleer de frontend geconfigureerd kan worden.

```
frontend:
  image: ghcr.io/shuffle/shuffle-frontend:latest
  container_name: shuffle-frontend
  hostname: shuffle-frontend
  ports:
    - "${FRONTEND_PORT}:80"
    - "${FRONTEND_PORT_HTTPS}:443"
  networks:
    - shuffle
  environment:
    - BACKEND_HOSTNAME=${BACKEND_HOSTNAME}
  restart: unless-stopped
  depends_on:
    - backend
```

Schermafbeelding van frontendconfiguratie in docker-compose.yml

Backend

De backend van Shuffle heeft als functie het centrale component te zijn in het platform. Het is opgebouwd als een REST API die inkomende verzoeken van de frontend of externe systemen afhandelt. Het stuurt hierbij de verschillende interne componenten aan zoals de database, workflow engine en Orborus. Wanneer er een HTTP-verzoek binnenkomt, voert de backend de volgende stappen uit:

1) Authenticatie en autorisatie worden eerst gecontroleerd. Er wordt zo bepaald of het verzoek geldig is en of de gebruiker toegang heeft tot de gevraagde actie.
2) Daarna wordt het verzoek verwerkt**.** Afhankelijk van het type verzoek kunnen er bijvoorbeeld workflows worden gestart, logs worden opgehaald, of gegevens worden aangepast.
3) Indien nodig wordt een taak aangemaakt en gestart via Orborus, dat verantwoordelijk is voor het uitvoeren van taken binnen workflows.
4) Tot slot stuurt de backend een HTTP-statuscode en eventueel aanvullende gegevens terug naar de gebruiker om aan te geven wat het resultaat van het verzoek is.

Op de onderstaande schermafbeelding kunt u de configuratie van de backend zien in de docker-compose.yml. Hieronder licht ik kort toe waarvoor de verschillende opties dienen:

- ‘image’: De Docker image waarop de backend gebaseerd is, Shuffle heeft zijn eigen backend image die hiervoor gebruikt wordt.
- ‘container_name’: De naam die gegeven wordt aan de Docker container voor makkelijk de container te herkennen.
- ‘hostname’: De hostnaam die gegeven wordt aan de Docker container voor interne communicatie. De naam komt uit het .env-bestand.
- ‘ports’: De poort die gebruikt wordt om met de backend te verbinden. Dit kan zelf gekozen worden door het aan te passen in het .env-bestand, maar is standaard 5001.
- ‘networks’: Het interne netwerk waarmee de backend verbonden is en communiceert met de andere componenten.
- ‘volumes’: Maakt mappen van de hostmachine beschikbaar binnen de container. De container kan hierdoor bestanden lezen of opslaan op een plek buiten de container.
- ‘env_file’: Laadt de omgevingsvariabelen, die in het .env-bestand zitten, in.
- ‘environment’: Omgevingsvariabelen waar de paden van de shuffle-apps en shuffle-files ingegeven worden. Zo weet de backend waar deze te vinden zijn.
- ‘restart’: Docker herstart de container automatisch als het stopt, tenzij dit door de gebruiker gebeurt.

```yaml
backend:
  image: ghcr.io/shuffle/shuffle-backend:latest
  container_name: shuffle-backend
  hostname: ${BACKEND_HOSTNAME}
  # Here for debugging:
  ports:
    - "${BACKEND_PORT}:5001"
  networks:
    - shuffle
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock
    - ${SHUFFLE_APP_HOTLOAD_LOCATION}:/shuffle-apps:z
    - ${SHUFFLE_FILE_LOCATION}:/shuffle-files:z
  env_file: .env
  environment:
    #- DOCKER_HOST=tcp://docker-socket-proxy:2375
    - SHUFFLE_APP_HOTLOAD_FOLDER=/shuffle-apps
    - SHUFFLE_FILE_LOCATION=/shuffle-files
  restart: unless-stopped
```

Schermafbeelding van backendconfiguratie in docker-compose.yml

Orborus
Orborus is het uitvoerende component binnen Shuffle en heeft als taak het afhandelen van taken binnen workflows. Het ontvangt instructies van de backend over hoe de workflow uitgevoerd moet worden. Het volgt deze instructies stap voor stap op om het gewenste resultaat te bieden aan de gebruiker.
Orborus draait op een aparte container om schaalbaarheid en betrouwbaarheid te garanderen. Het is hierdoor mogelijk om meerdere workflows tegelijk in parallel uit te voeren, wat extra functionaliteiten biedt voor de gebruiker.

Op de onderstaande schermafbeelding kunt u de configuratie van Orborus zien in 'docker-compose.yml'. Hieronder licht ik kort toe waarvoor de verschillende opties dienen:

- 'image': De Docker image waarop Orborus gebaseerd is, Shuffle heeft zijn eigen Orborus image die hiervoor gebruikt wordt.
- 'container_name': De naam die gegeven wordt aan de Docker container voor makkelijk de container te herkennen.
- 'hostname': De hostnaam die gegeven wordt aan de Docker container voor interne communicatie..
- 'networks': Het interne netwerk waarmee de backend verbonden is en communiceert met de andere componenten.
- 'volumes': Maakt mappen van de hostmachine beschikbaar binnen de container. De container kan hierdoor bestanden lezen of opslaan op een plek buiten de container.
- 'environment': Hierin staan de omgevingsvariabelen die gebruikt worden in het .env-bestand. In Bijlage x staan de verschillende opties uitgelegd.
- 'env_file': Laadt de omgevingsvariabelen, die in het .env-bestand zitten, in.
- 'restart': Docker herstart de container automatisch als het stopt, tenzij dit door de gebruiker gebeurt.
- 'security_opt': Bepaalt welke systeemaanroepen uitgevoerd mogen worden. Orborus wordt hier niet in beperkt en mag ze allemaal uitvoeren.

```yaml
orborus:
  image: ghcr.io/shuffle/shuffle-orborus:latest
  container_name: shuffle-orborus
  hostname: shuffle-orborus
  networks:
    - shuffle
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock
  environment:
    - SHUFFLE_APP_SDK_TIMEOUT=300
    - SHUFFLE_ORBORUS_EXECUTION_CONCURRENCY=7 # The amount of concurrent executions Orborus can handle.
    #- DOCKER_HOST=tcp://docker-socket-proxy:2375
    - ENVIRONMENT_NAME=Shuffle
    - ORG_ID=Shuffle
    - BASE_URL=http://${OUTER_HOSTNAME}:5001
    - DOCKER_API_VERSION=1.40
    - HTTP_PROXY=${HTTP_PROXY}
    - HTTPS_PROXY=${HTTPS_PROXY}
    - SHUFFLE_PASS_WORKER_PROXY=${SHUFFLE_PASS_WORKER_PROXY}
    - SHUFFLE_PASS_APP_PROXY=${SHUFFLE_PASS_APP_PROXY}
    - SHUFFLE_STATS_DISABLED=true
    - SHUFFLE_LOGS_DISABLED=true
    - SHUFFLE_SWARM_CONFIG=run
    - SHUFFLE_WORKER_IMAGE=ghcr.io/shuffle/shuffle-worker:latest
  env_file: .env
  restart: unless-stopped
  security_opt:
    - seccomp:unconfined
```

Schermafbeelding van Orborusconfiguratie in docker-compose.yml

Opensearch
Opensearch is de centrale database binnen Shuffle waar gegevens uit workflows opgeslagen, doorzocht en geanalyseerd worden. Deze gegevens kunnen van alles zijn zoals logs van uitgevoerde acties, foutmeldingen en status van workflows. Door het gebruik van Opensearch kunnen gebruikers makkelijk belangrijke data ophalen en terug te vinden wat er tijdens de workflows gebeurd is.
Opensearch draait net als de andere componenten in een aparte container. Dit maakt het makkelijker om troubleshooting uit te voeren en geeft extra schaalbaarheid en flexibiliteit voor de gebruiker.

Op de onderstaande schermafbeelding kunt u de configuratie van Opensearch zien in 'docker-compose.yml'. Hieronder licht ik kort toe waarvoor de verschillende opties dienen:

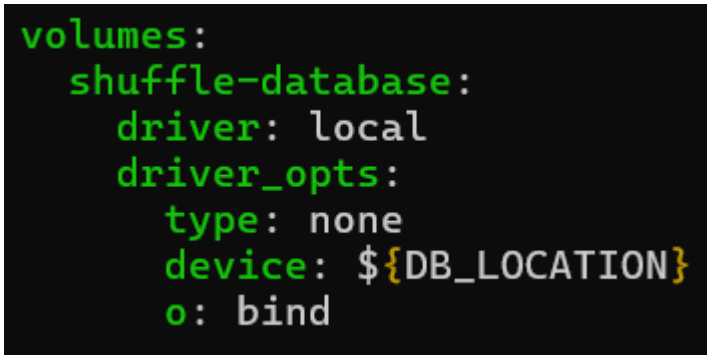
- 'image': De Docker image waarop Opensearch gebaseerd is, Shuffle heeft zijn eigen Opensearch image die hiervoor gebruikt wordt.
- 'hostname': De hostnaam die gegeven wordt aan de Docker container voor interne communicatie.
- 'container_name': De naam die gegeven wordt aan de Docker container voor makkelijk de container te herkennen.
- 'environment': Hierin staan de omgevingsvariabelen die gebruikt worden in het .env-bestand. In Bijlage x staan de verschillende opties uitgelegd.
- 'ulimits:memlock': Stelt in hoeveel RAM-geheugen gelocked mag worden. Dit houdt in dat het niet naar het wisselgeheugen wordt verplaatst. Door dit op -1 te zetten, betekent het dat dit geen limiet heeft. 'soft' geeft de standaardlimiet weer en 'hard' de maximumlimiet.
- 'ulimits:nofile': Stelt in hoeveel open bestanden en sockets Opensearch mag hebben. 'soft' betekent opnieuw de standaardlimiet en 'hard' de maximumlimiet.
- 'volumes': Maakt mappen van de hostmachine beschikbaar binnen de container. De container kan hierdoor bestanden lezen of opslaan op een plek buiten de container.
- 'ports': De poorten die gebruikt worden voor Opensearch.
- 'networks': Het interne netwerk waarmee de backend verbonden is en communiceert met de andere componenten.
- 'restart': Docker herstart de container automatisch als het stopt, tenzij dit door de gebruiker gebeurt.

```yaml
opensearch:
  image: opensearchproject/opensearch:2.19.1
  hostname: shuffle-opensearch
  container_name: shuffle-opensearch
  environment:
    - "OPENSEARCH_JAVA_OPTS=-Xms2048m -Xmx2048m" # minimum and maximum Java heap size, recommend setting both to 50% of system RAM
    - bootstrap.memory_lock=true
    - DISABLE_PERFORMANCE_ANALYZER_AGENT_CLI=true
    - cluster.initial_master_nodes=shuffle-opensearch
    - cluster.routing.allocation.disk.threshold_enabled=false
    - cluster.name=shuffle-cluster
    - node.name=shuffle-opensearch
    - node.store.allow_mmap=false
    - discovery.seed_hosts=shuffle-opensearch
    - OPENSEARCH_INITIAL_ADMIN_PASSWORD=${SHUFFLE_OPENSEARCH_PASSWORD}
  ulimits:
    memlock:
      soft: -1
      hard: -1
    nofile:
      soft: 65536
      hard: 65536
  volumes:
    - shuffle-database:/usr/share/opensearch/data:z
  ports:
    - 9200:9200
  networks:
    - shuffle
  restart: unless-stopped
```

Schermafbeelding van Opensearchconfiguratie in 'docker-compose.yml'

Volumes
Volumes worden gebruikt om data persistent op te slaan. Dit betekent dat deze data ook als de container stopt, blijft bestaan. Voor veel gevoelige data die niet verloren mag gaan, is dit de perfecte opslagplaats. Voor Shuffle is er het volume 'Shuffle-database' aangemaakt. Deze kan aangesproken worden in andere delen van het docker-compose bestand om bestanden in op te slaan.
Op onderstaande afbeelding kunt u zien hoe dit geconfigureerd is in 'docker-compose.yml'. Hieronder licht ik kort toe wat de verschillende opties doen:

- 'driver': Hier wordt aangegeven waar de bestanden opgeslagen worden. In dit geval is dit lokaal op het hostsysteem.
- 'driver_opts': Extra opties die toegevoegd kunnen worden aan het volume.
- 'driver_opts.type': Geeft aan welk specifiek type bestandsysteem wordt gebruikt.
- 'driver_opts.device': Geeft aan welke map op het hostsysteem gebruikt moet worden om de bestanden in op te slaan.
- 'driver_opts.o': Geeft aan dat het volume een bind-mount is. Dit wil zeggen dat de map op het hostsysteem gekoppeld wordt aan de containers.



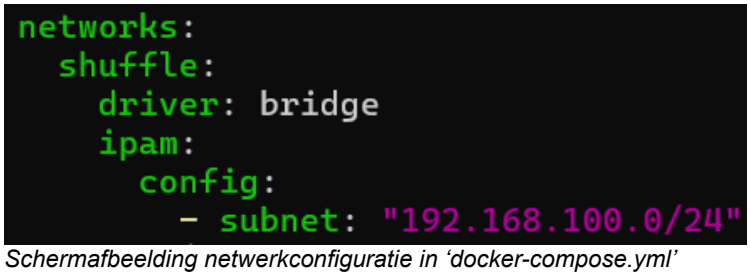
Schermafbeelding van volumeconfiguratie in 'docker-compose.yml'

Networks
Ten slotte is er nog het netwerk dat geconfigureerd moet worden. Dit zet een virtueel netwerk op dat de verschillende componenten gebruiken om met elkaar te verbinden. Het netwerk krijgt de naam Shuffle en dit wordt ingegeven bij de componenten zodat ze weten met welk netwerk ze verbinden.

Dit is ook het deel dat voor het meeste problemen heeft gezorgd bij het opzetten van Shuffle in mijn SOC. De installatie van Shuffle wou niet succesvol lukken omdat er ergens een error zat die ik niet kon vinden. Na meerdere dagen afspeuren van alle verschillende redenen, ben ik uiteindelijk naar de routes gaan kijken die Docker had opgesteld. Deze routes bepalen hoe netwerkverkeer zijn eindbestemming bereikt door als bestemming een IP-subnet op te geven. Enkele van deze routes gebruikten hetzelfde IP-subnet die ook in het netwerk van het MPI Oosterlo gebruikt werden. Daarom verloor ik telkens verbinding met mijn server, omdat de route naar mijn VLAN overschreven werd. Door het IP-subnet aan te passen, was het probleem opgelost.

Op onderstaande schermafbeelding kunt u de configuratie zien van het netwerk in 'docker-compose.yml'. Hieronder licht ik kort toe wat de verschillende opties doen:

- 'driver': Het netwerkmodel dat gebruikt wordt in Docker. In dit geval is het een 'bridge' waarbij er een virtueel netwerk wordt opgesteld dat met de buitenwereld kan communiceren via de host.
- 'ipam.config.subnet': Bepaalt het IP-subnet dat gebruikt wordt voor het virtueel netwerk.



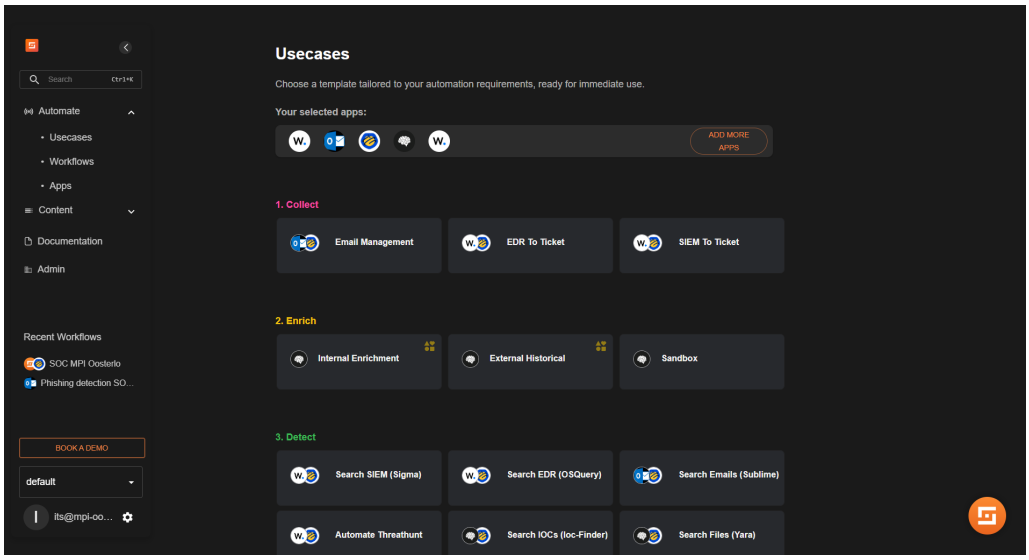
Schermafbeelding netwerkconfiguratie in 'docker-compose.yml'

DASHBOARD

Nu Shuffle volledig geïnstalleerd is, kan de workflow opgesteld worden. Hiervoor is er enkel een account nodig, wat snel geregistreerd kan worden bij Shuffle zelf. Na een succesvolle aanmelding bij Shuffle komen we op het dashboard, dat op onderstaande schermafbeelding zichtbaar is.

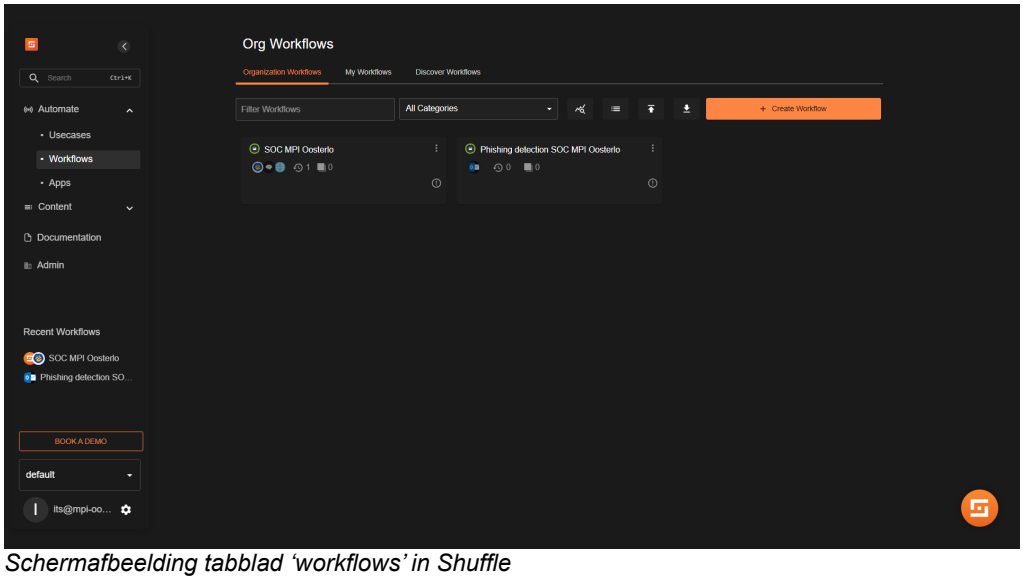
Standaard komen we uit op het tabblad ‘usecases’. Dit geeft ons enkele voorbeelden van mogelijke workflows die opgesteld kunnen worden. Links staat er een uitklapmenu waarin er nog meer opties zijn. De belangrijkste hiervan is het deel ‘Automate’. Dit heb ik het meeste in mijn SOC gebruikt. Er zijn drie tabbladen te vinden, die elk een deel vormen van het automatische proces van Shuffle.

Het eerste tabblad zijn de ‘usecases’. Dit zijn voorbeelden van workflows die al door Shuffle of andere gebruikers zijn opgezet. Het is hier mogelijk te filteren op gebruikte tools en hier een workflow mee opzetten. Deze workflows zijn onderverdeeld op de taken die ze aanbieden, zoals bijvoorbeeld het verzamelen van data of het detecteren van bedreigingen.



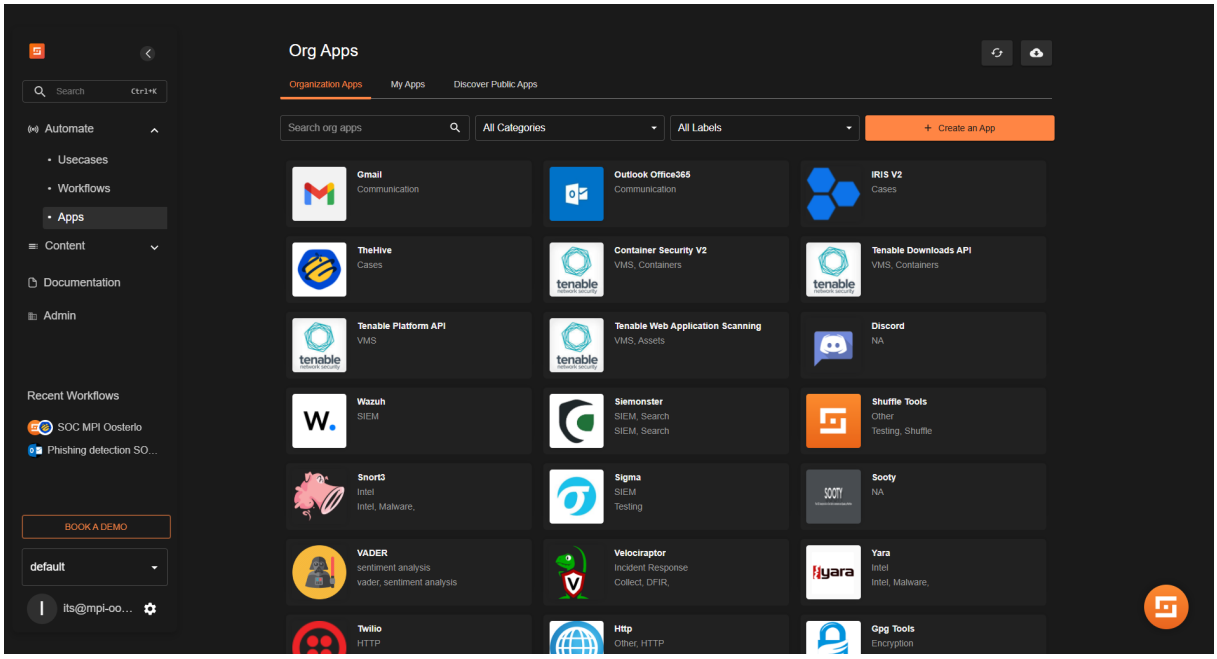
Schermafbeelding tabblad ‘usecases’ in Shuffle

Het tweede zijn de ‘workflows’. Hierin staan de workflows die de gebruiker zelf aanmaakt. Op onderstaande schermafbeelding kunt u mijn workflows zien die ik voor het MPI Oosterlo gemaakt heb. Hier ga ik in het volgende deel dieper op in.



Schermafbeelding tabblad ‘workflows’ in Shuffle

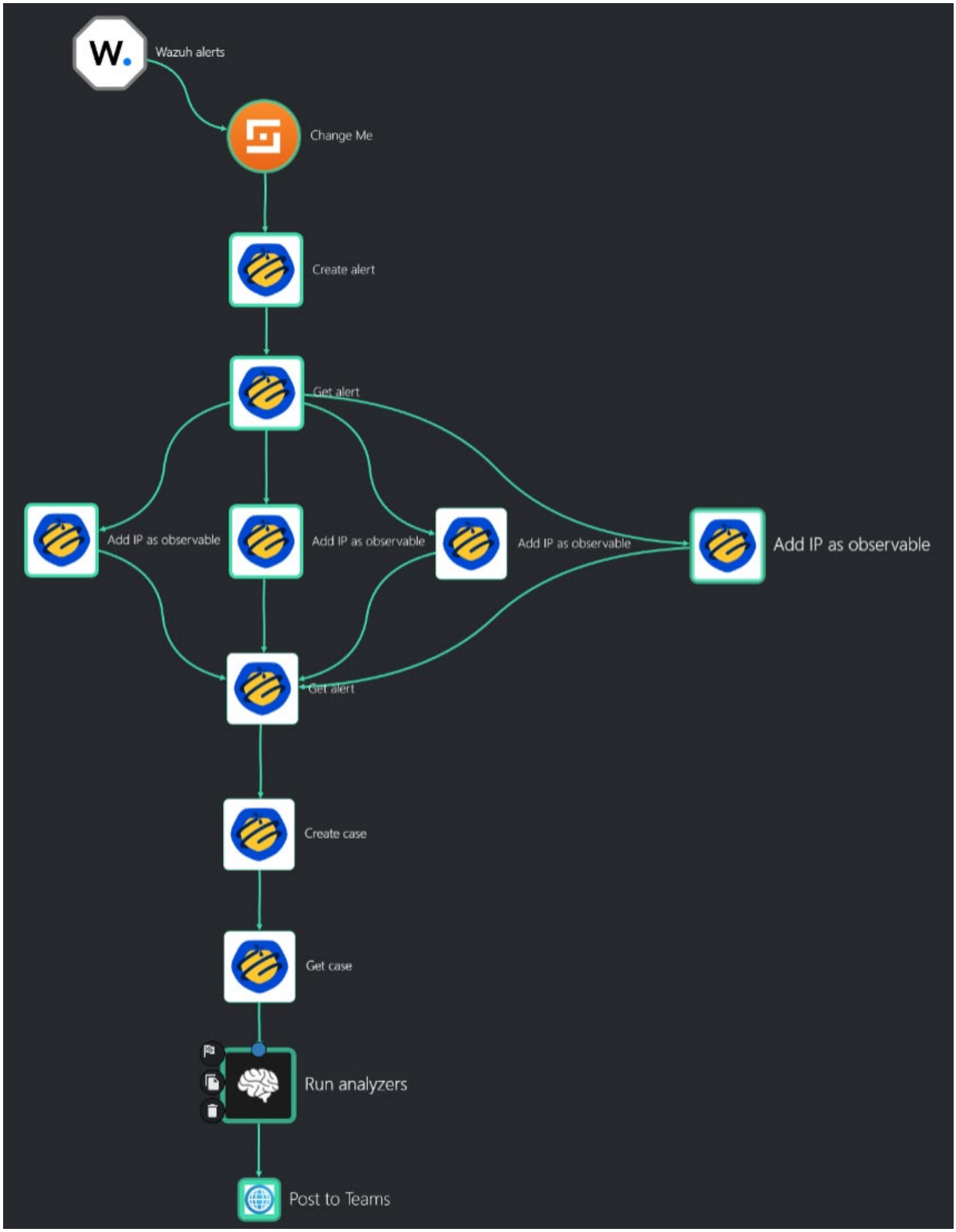
Het derde en laatste tabblad is ‘apps’. Shuffle heeft verschillende tools waarmee een automatische workflow opgezet kan worden. Deze tools worden in een app gezet die alle nodige en optionele velden bevat. Zo krijgt de gebruiker een visuele voorstelling van hoe een API-verzoek naar de tool eruit zou zien. Op de schermafbeelding hieronder kunt u de verschillende apps zien die Shuffle standaard aanbiedt. Deze zijn vaak door mensen in de gemeenschap gemaakt en officieel toegevoegd door Shuffle.
Mocht er toch de nood zijn om bij een applicatie een eigen app te maken, is dit ook mogelijk. Shuffle apps zijn in Python geschreven met OpenAPI or Swagger. De gebruiker kan zelf de code schrijven of de App laten maken door gebruik te maken van de GUI.



Schermafbeelding tabblad ‘apps’ in Shuffle

WORKFLOW 'SOC MPI OOSTERLO'

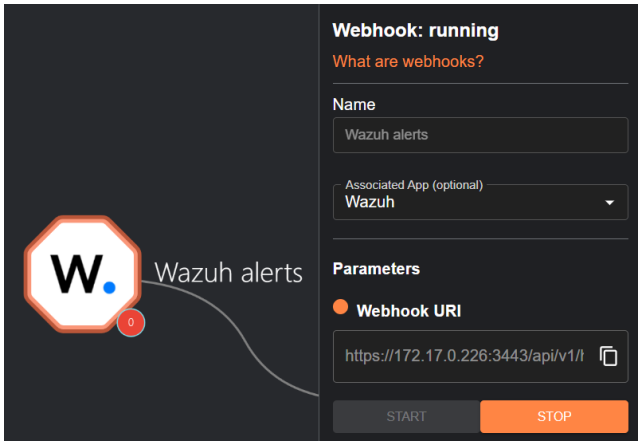
Op onderstaande schermafbeelding kunt u mijn volledige workflow zien. Dit vormt de centrale component die alerts van Wazuh ophaalt via, deze naar TheHive stuurt voor er een case van te maken, de case te analyseren en een melding te maken in Teams. In de volgende delen ga ik elk deel van de workflow uitleggen en welke functie dit heeft in mijn SOC.



Schermafbeelding workflow 'SOC MPI Oosterlo'

Wazuh alerts

Om de workflow in gang te zetten, moet er een trigger zijn die de start maakt. Zoals daarstraks al besproken zijn er hiervoor vier opties: een webhook, een schedule, een subflow en user input. Omdat Wazuh op onregelmatige momenten een alert genereerd en deze direct onderzocht moet worden, is de webhook de beste keuze voor ons. Deze wordt aangesproken door Wazuh wanneer er een alert aan bepaalde voorwaarden voldoet en zet dan de workflow in gang. De webhook is een URI die door Shuffle gegeneerd wordt om vanuit verschillende tools met Shuffle te verbinden. Op de schermafbeelding hieronder ziet u hoe deze webhook opgezet is in Shuffle.



Schermafbeelding webhook Wazuh in Shuffle

Om deze integratie te laten werken, zijn er nog twee bestanden nodig die zich bevinden op de Github van Wazuh. Dit zijn 'custom-shuffle' en 'custom-shuffle.py'. Er moet niet veel aan deze bestanden veranderd worden, enkel in 'custom-shuffle.py' is er een kleine aanpassing nodig. Hier wordt 'verify=False' toegevoegd aan de 'send_msg' functie, omdat Shuffle met HTTPS werkt via self-signed certificaten. Deze kunnen niet gecontroleerd worden door Wazuh en zou de integratie geblokt worden. Door de controle af te zetten, werkt de integratie wel. Op de schermafbeelding hieronder ziet u de aanpassing in de functie 'send_msg' in 'custom-shuffle.py'. De volledige bestanden zijn te groot om hier te zetten, maar kunt u vinden in Bijlage x.

```
def send_msg(msg, url):
    debug("# In send msg")
    headers = {'content-type': 'application/json', 'Accept-Charset': 'UTF-8'}
    res = requests.post(url, data=msg, headers=headers, verify=False)
    debug("# After send msg: %s" % res)
```

Schermafbeelding functie 'send_msg' uit 'custom_shuffle.py'

Nu alles klaar staat voor alerts naar Shuffle te sturen, moet enkel 'ossec.conf' nog aangepast worden. Wazuh voorziet hiervoor een blok genaamd 'integration' in het 'ossec.conf'-bestand. Deze kunt u zien op de schermafbeelding hieronder. Het blok 'integration' staat online in de documentatie van Wazuh en kan naar 'ossec.conf' gekopieerd worden. Nu kan de integratie opgezet worden door enkele opties aan te passen. Hieronder licht ik kort toe wat de opties doen:

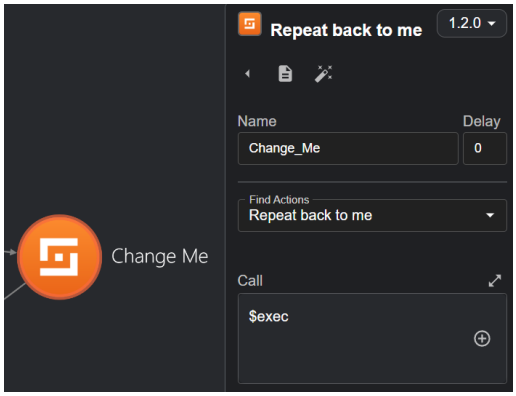
- 'name': De naam die gegeven wordt aan de integratie, moet overeenkomen met de naam van de integratiebestanden.
- 'hook_url': De webhook waarmee de integratie opgezet wordt.
- 'level': Het meldingsniveau van de alerts die doorgestuurd worden met de webhook.
- 'alert_format': In welk formaat de alerts doorgestuurd worden.

```
<integration>
  <name>custom-shuffle</name>
  <hook_url>https://172.17.0.226:3443/api/v1/hooks/webhook_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx </hook_url>
  <level>12</level>
  <alert_format>json</alert_format>
</integration>
```

Schermafbeelding blok 'integration' uit 'ossec.conf'

Repeat back to me

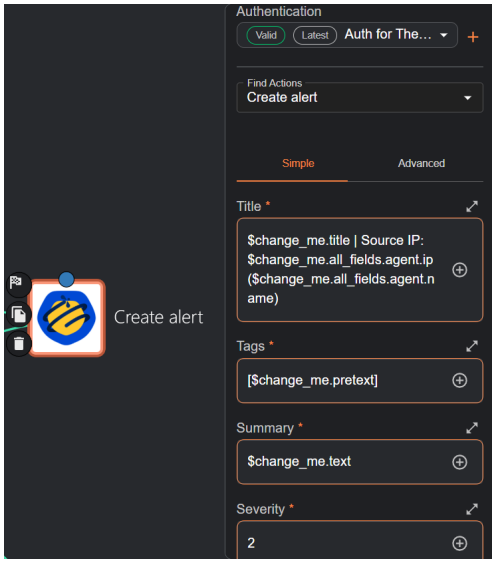
Nu er alerts binnenkomen op Shuffle, kunnen we met deze gegevens aan de slag gaan. Met Shuffle is het niet mogelijk om gegevens, die uit een webhook komen, later terug aan te spreken. Als er dus iets moet gebeuren met de alerts, zoals bijvoorbeeld een observable opzetten, kan dit niet meer opgeroepen worden. Daarom is de functie ‘Repeat back to me’ toegevoegd aan de workflow. Dit is een functie die bij in de app Shuffle-tools zit en die standaard toegevoegd wordt aan een nieuwe workflow.
De functie ‘Repeat back to me’ herhaalt de laatste gegevens die het binnenkrijgt. In ons geval gaat dit de data zijn die binnenkomt met de webhook. Deze gegevens zitten in de ‘runtime argument’ wat aangeroepen wordt met ‘$exec’. Op de schermafbeelding hieronder kunt u de app zien in Shuffle.



Schermafbeelding ‘Repeat back to me’ in Shuffle

Create alert

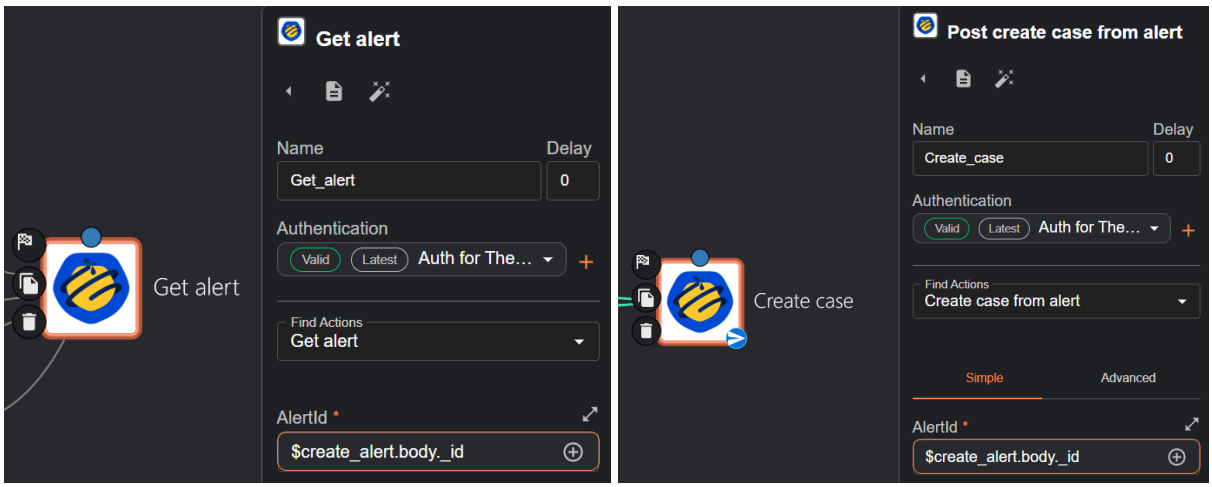
Nu de gegevens van de Wazuh alert opgehaald kunnen worden, kan er een alert gemaakt worden in TheHive. Dit gebeurt via de app ‘TheHive’ die met een API-call een nieuwe alert kan aanmaken. Hiervoor moet er eerst een verbinding worden opgezet met de server van TheHive. Shuffle heeft hiervoor een veld genaamd ‘authentication’. Hierin worden de benodigde zaken getoond voor een connectie op te stellen met de service. Voor TheHive is dit de URL van de server en een API-key. De API-key is afkomstig van een gebruiker in een organisatie in TheHive, maar hier kom ik straks in dat deel op terug. Shuffle slaat deze verbinding op zodat het gebruikt kan worden wanneer de app van TheHive opnieuw gebruikt wordt.
Nu kan de alert voor TheHive gemaakt worden. Hiervoor zijn vier zaken nodig: een titel, een tag, een samenvatting en de ernst. Om deze zaken in te vullen, worden gegevens vanuit de Wazuh alert gebruikt. Zoals u in de schermafbeelding hieronder kunt zien, worden variabelen gebruikt die direct uit de vorige app ‘Repeat back to me’ komen. Zo worden de gegevens dynamisch aangepast op basis van de binnenkomende alerts. Enkel de ernst staat op een vaste waarde, omdat Shuffle hier syntax-errors geeft. In Bijlage x kunt u van een alert zien hoe de verschillende velden ingevuld worden.



Schermafbeelding ‘Create alert’ uit Shuffle

Get alert/case

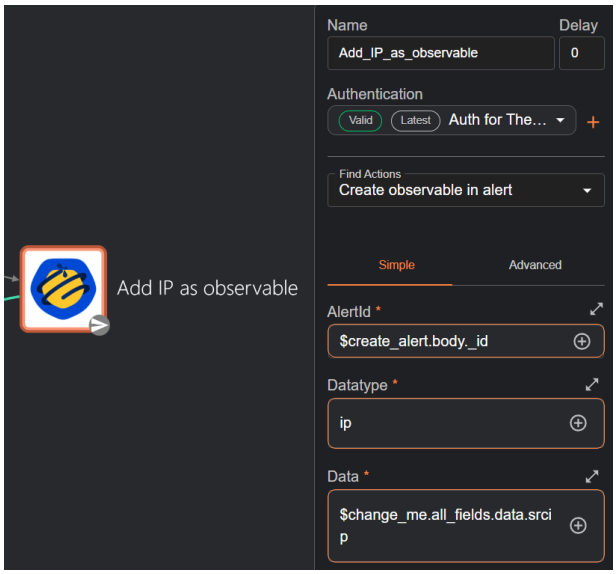
Om troubleshooting in de workflow makkelijker te maken, is het handig om te zien wat de verschillende acties genereren. Daarvoor zijn de 'Get alert' of 'Get case' functies perfect. Hiermee kan de alert of case die net aangemaakt is, opgehaald en onderzocht worden. Zo kunnen fouten in de workflow snel opgespoord en op gereageerd worden.
Op de schermafbeelding hieronder ziet u de 'Get alert' functie. Deze werkt door de alert-ID op te vragen. De 'Get case' functie hanteert hetzelfde principe en kunt u vinden in Bijlage x.



Schermafbeeldingen van 'Get alert' en 'Get case' uit Shuffle

Add IP as observable

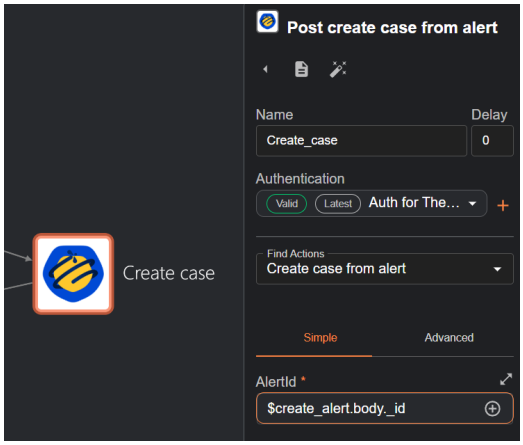
Nu de alerts aangemaakt worden in TheHive, is het tijd om hier data aan toe te voegen dat onderzocht kan worden. Hiervoor wordt er gebruik gemaakt van observables. Deze worden in TheHive toegevoegd in een apart tabblad binnen de alert, maar hier kom ik straks op terug.
Om een observable toe te voegen, wordt de funtie 'Create observable in alert' gebruikt. Hiervoor zijn er enkele zaken nodig: een alertID, het datatype en de data. Deze worden opnieuw ingevuld met variabelen zoals ook met het maken van een alert. Enkel het datatype is een statisch gegeven.
In mijn SOC heb ik een IP-adres toegevoegd als observable. Omdat hier het probleem opkomt van Wazuh die verschillende indexen geeft hieraan, heb ik een andere oplossing moeten zoeken. Daarom heb ik drie verschillende manieren gebruikt om een IP-adres toe te voegen. Op deze manier kunnen we er zeker van zijn dat het toegevoegd wordt.
Op de schermafbeelding hieronder ziet u de 'Create observable in alert' functie. Deze werd gebruikt voor alerts uit Linuxendpoints. De andere apps voor Windows kunt u vinden in Bijlage x.



Schermafbeelding 'Create observable in alert' uit Shuffle

Create case

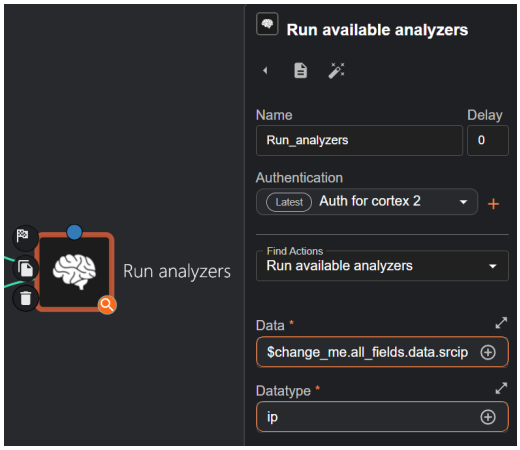
Nu we een alert hebben waar een observable aan toegevoegd is, kan er een case gemaakt worden van de alert. Hiervoor wordt de functie ‘Create case from alert’ gebruikt. Om dit te laten werken, is enkel het alert-ID nodig. Hiermee kan TheHive zijn eigen alert ophalen en gebruiken om een case mee te maken. Aan deze case worden dan ook de observables toegevoegd uit de vorige stap.
Cases zijn handig omdat ze meer functionaliteiten bieden voor de gebruiker en organisatie. In mijn uitleg over TheHive ga ik hier dieper op in.
Op de schermafbeelding hieronder ziet u de ‘Create case from alert’ functie.



Schermafbeelding ‘create case from alert’ uit Shuffle

Run analyzers

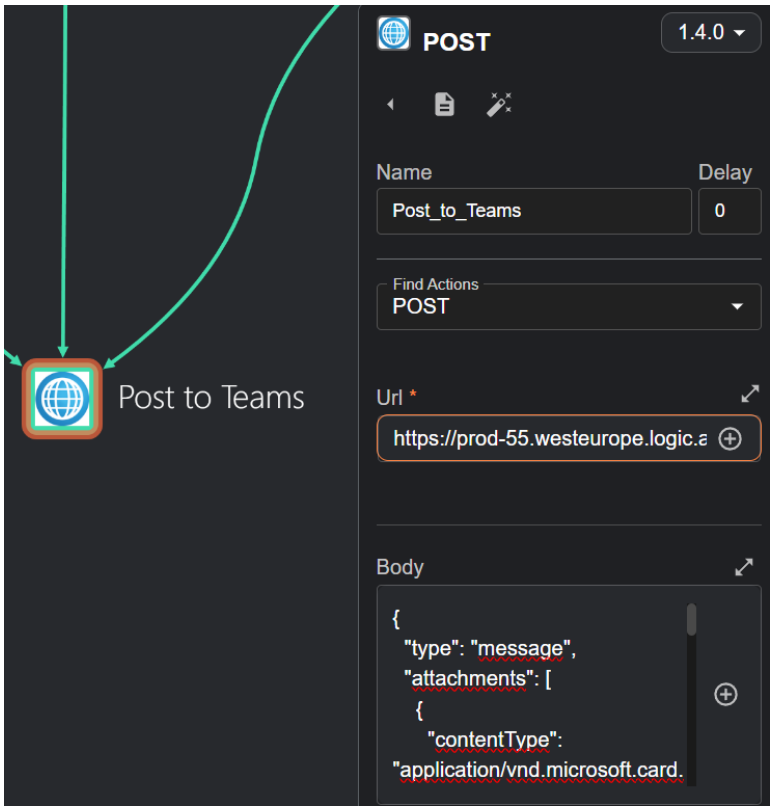
Nu er een case met observables is gemaakt, kunnen deze observables geanalyseerd worden. Hiervoor wordt de app van ‘Cortex’ gebruikt met de functie ‘Run available analyzers’. Hiermee gaat Cortex alle mogelijke analyzers oproepen om onderzoek te doen naar het IP-adres.
Om deze analyse mogelijk te maken, moet er eerst een verbinding zijn met Cortex. Hiervoor is, net zoals met TheHive, de URL van de server en de API-key van een gebruiker in een organisatie nodig. Hier kom ik opnieuw later op terug in het deel over Cortex.
Om de analyzers te laten werken, moeten ze eerst weten wat er geanalyseerd moet worden. Shuffle heeft hiervoor twee zaken nodig: de data en het datatype. Deze zijn zichtbaar op onderstaande schermafbeelding. Normaal zou hiervoor de observable gebruikt kunnen worden, deze heeft namelijk beide al gedefinieerd, maar in mijn SOC komen de observables niet bij in de case op Shuffle te staan. Het is dus niet mogelijk om deze op te halen met een variabele uit de case. Daarom heb ik dezelfde oplossing toegepast die ik ook bij het toevoegen van mijn observables heb gebruikt. Er zijn drie apps die de IP-adressen ophalen van alerts uit Linux- en Windowsendpoints en in de data steken. Het datatype staat statisch op IP-adres ingesteld.
De analyzers in Cortex kunnen nu deze informatie gebruiken om hun analyse uit te voeren. De resultaten hiervan zijn in Cortex zichtbaar, maar hier kom ik straks in het deel over Cortex op terug.



Schermafbeelding ‘Run available analyzers’ uit Shuffle

Post to Teams

Tenslotte is er nog de melding die aangemaakt wordt op Teams. Een SOC kan natuurlijk niet volledig automatisch zijn als er nergens een melding is van een nieuwe alert. Hiervoor zijn verschillende opties mogelijk zoals Outlook, Teams en Discord. Zoals eerder al vermeld heb ik hier de beslissing genomen om voor Teams te gaan.
Om een integratie met Teams te maken, moet er eerst een connectie zijn. Shuffle biedt zelf geen apps aan die connectie te maken, maar gelukkig Teams wel. Hier kan er een webhook opgesteld worden waarmee Shuffle kan verbinden. Deze wordt aangemaakt in Teams zelf en is gelinkt aan een chat, maar hier ga ik in het deel over Teams verder op in.
Om de webhook aan te kunnen spreken, wordt er gebruik gemaakt van een POST-aanvraag. Dit is een aanvraag die dient om nieuwe informatie, of in dit geval een nieuw bericht, op te sturen naar een URL. Shuffle vraagt dan ook om een URL (de webhook) en een body toe te voegen. In de body wordt het bericht geplaatst dat naar Teams verstuurd moet worden. Dit moet in de juiste structuur zijn dat bepaald wordt door Microsoft en te vinden is in de Power Automate Portal.
Op onderstaande schermafbeelding kunt u zien hoe de app er in Shuffle uitziet. Het bericht bevat de titel van de alert, het IP-adres dat de alert gegenereerd heeft en de hostname hiervan. De volledige body kunt u vinden in Bijlage x.



Schermafbeelding 'Post to teams' uit Shuffle

3.2.2. TheHive

SETUP

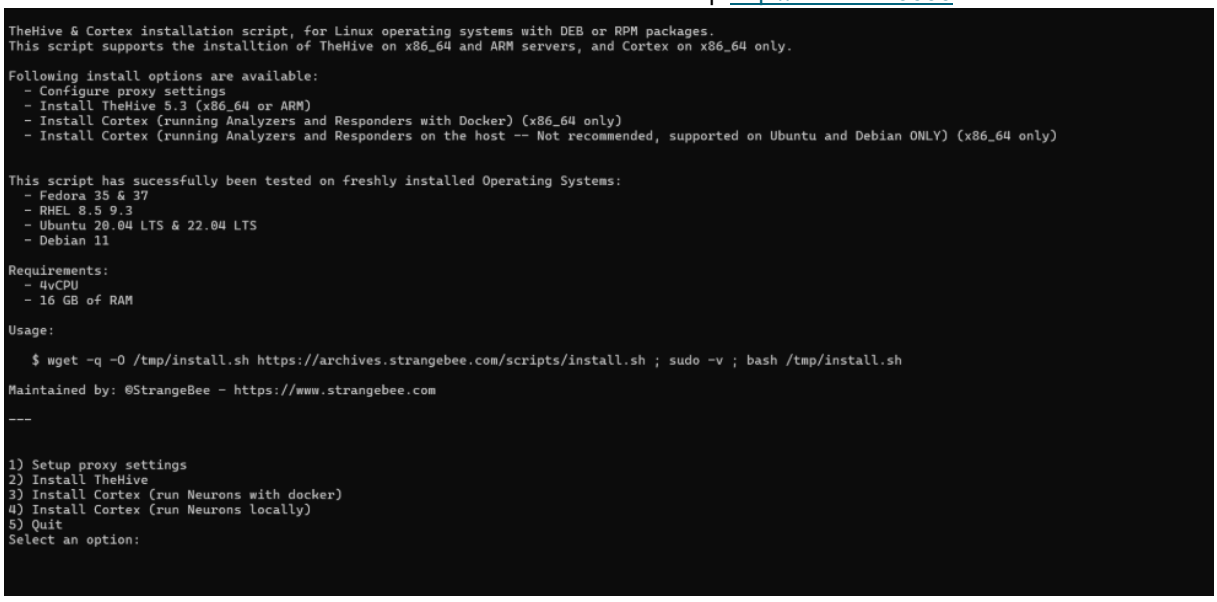
TheHive bestaat uit vijf hoofdcomponenten die nodig zijn om een goede instantie op te kunnen zetten. Hieronder licht ik kort toe welke dit zijn en wat ze doen:

- Java Virtual Machine: Zorgt ervoor dat programma's die geschreven zijn in Java, uitgevoerd kunnen worden op het systeem.
- Apache Cassandra: Een NoSQL-database die gebruikt wordt om cases, alerts en observables op te slaan.
- Elasticsearch: Wordt gebruikt voor de database te doorzoeken en te indexeren.
- File storage: Zorgt ervoor dat bestanden opgeslagen kunnen worden op het lokale bestandssysteem of op een gedeeld opslagvolume
- TheHive: De hoofdapplicatie die via een webinterface de mogelijkheid biedt om cases en alerts te beheren en onderzoeken. Het staat toe om vlot samen te werken met andere leden van de organisatie.

a) Installatie

De componenten kunnen afzonderlijk geïnstalleerd worden, of via een installatiescript. Bij meerdere pogingen tot handmatige installatie traden er telkens fouten op. Daarom is ervoor gekozen om het installatiescript te gebruiken, wat wel succesvol bleek. Sindsdien is deze methode in gebruik gebleven en is de handmatige installatie niet meer getest.

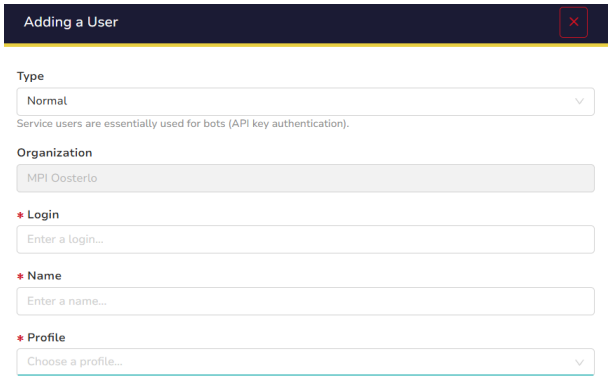
Het installatiescript kan makkelijk van de site van TheHive gehaald worden. Het voorziet in de documentatie een commando waarmee dit opgehaald kan worden. Het enige wat dan nog rest is om het script uit te voeren. Het script geeft vijf mogelijkheden die uitgevoerd kunnen worden. Deze kunt u zien op onderstaande schermafbeelding. Er wordt gekozen voor optie 2 om de installatie van TheHive te starten. Na de installatie is de webinterface van TheHive beschikbaar op http://IP-adres:9000.



b) Configuratie

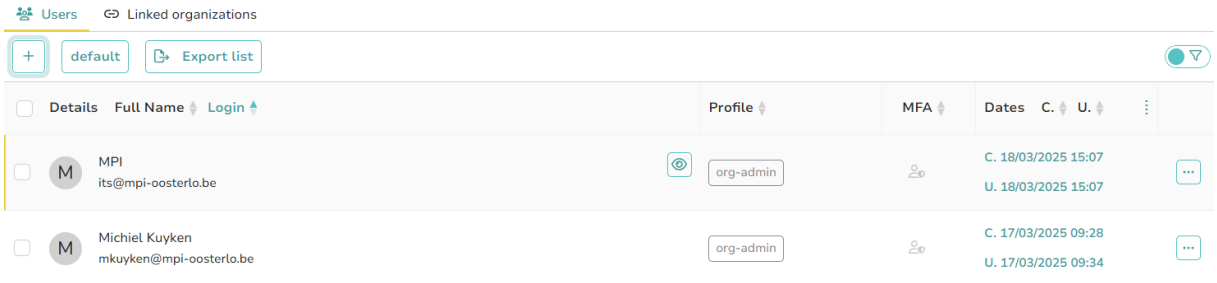
Zoals eerder al vermeld werkt TheHive met organisaties en gebruikers binnen deze organisaties. Het doet dit zodat grote bedrijven verschillende organisaties kunnen opzetten per team. Het MPI Oosterlo heeft een klein IT-team, dus volstaat het om slechts één organisatie hiervoor aan te maken. Dit kan in het adminportaal waar standaard toegang tot is met gebruikersnaam en wachtwoord admin. Het wordt geadviseerd om dit wachtwoord meteen aan te passen na de eerste inlog.

Nu er toegang is tot het adminportaal, kan er een nieuwe organisatie gemaakt worden. Deze krijgt de naam MPI Oosterlo. In de organisatie kunnen gebruikers toegevoegd worden door op de plus te klikken onder ‘users’. Er opent dan het menu dat u op onderstaande schermafbeelding kan zien.
Er zijn verschillende mogelijkheden voor de nieuwe gebruiker. Eerst moet het type bepaald worden, dit kan ‘normal’ of ‘service’ zijn. Een ‘normal’-gebruiker is een standaardgebruiker die voor leden van het team gebruikt wordt. Een ‘serivce’-gebruiker is voor automatisatie te bevorderen. Deze gebruikers worden meestal enkel gebruikt voor API-keys en verbindingen op te zetten met andere services. De gebruikers in de organisatie MPI Oosterlo zijn allemaal type ‘normal’. Dit is omdat zij ook een API-key hebben, maar er slechts één verbinding opgezet moet worden naar Shuffle.
De nieuwe gebruiker moet een e-mailadres en naam krijgen. Hier is het e-mailadres gebruikt van de IT-dienst gebruikt en als naam ‘MPI’.
Als laatste moet er nog het profiel bepaald worden. Zoals eerder al vermeld zijn er drie verschillende profielen met elk hun eigen rechten. Het account ‘MPI’ krijgt org-admin als profiel zodat het toegang heeft tot alles. Eén account is in principe voldoende voor het MPI Oosterlo, maar hier kunnen ze in de toekomst extra account toevoegen moest dit nodig zijn.



Schermafbeelding van gebruiker toevoegen aan organisatie

Nu het account aangemaakt is, komt dit bij in de organisatie te staan. Er is nu de mogelijkheid om een wachtwoord toe te voegen en de API-key op te halen. Deze API-key wordt gebruikt voor authenticatie in tools zoals Shuffle, zoals eerder al vermeld in de uitleg van Shuffle. Dit kan door de gebruiker te bekijken met het oog dat langs het profiel komt te staan. Op onderstaande schermafbeelding kunt u zien hoe de lijst van gebruikers eruitziet.



Schermafbeelding lijst van gebruikers binnen de organisatie

Het laatste dat nog in orde moet worden gebracht, is de licentie van TheHive. Zoals eerder vermeld heeft TheHive een gratis community-versie die gebruikt gaat worden in het SOC. Dit biedt genoeg functionaliteiten voor het SOC. Er is hiervoor wel een account nodig op het licenseportaal van Strangebee, de overkoepelende organisatie van TheHive en Cortex. Na de eerste aanvraag moet dit elk jaar vervangen worden. De handleiding om dit te vervangen, kunt u vinden in Bijlage x.

DASHBOARD

a) Alerts
b) Cases

3.2.3. Cortex

a) Installatie
b) Configuratie
c) Analyzers

3.2.4. Teams

a) Configuratie

3.2.5. DFIR IRIS

3.3. Threat Intel

3.3.1. MISP

a) Installatie
b) Configuratie
c) Integratie
d) Events

Dit hoofdstuk en de volgende bevatten de beschrijving van het resultaat dat je hebt gemaakt: de functionaliteiten, de opzet van de toepassing, bepaalde principes die je hebt toegepast, welke patronen je geïmplementeerd hebt…
Deze beschrijving moet niet volledig zijn maar moet een goed idee geven van wat je hebt gemaakt. Kleine stukjes code ter illustratie, bepaalde schermen en hoe die werken, enz.… staan hier op hun plaats.

4. Besluit

Het besluit is een "afsluiting" waarmee het schrijfwerk wordt afgerond.
In het besluit vind je vaak:

- een korte recapitulatie of terugblik; de belangrijkste informatie wordt nog eens in de verf gezet
- conclusies, getrokken uit al wat voorafgaat
- een soort van evaluatie; vaak wordt er teruggekoppeld naar de doelstellingen die in het projectplan werden geformuleerd
- een vooruitblik naar de toekomst: adviezen, aanbevelingen, verwachtingen, waarschuwingen

LITERATUURLIJST

De **literatuurlijst** mag niet verward worden met de bibliografie. Die laatste wil een volledig overzicht geven van de belangrijkste publicaties over je thema. Dat is wel wat ambitieus voor dit document, waarvoor wel een literatuurlijst verwacht wordt. Je moet daarin een overzicht geven van alle geschreven en gesproken bronnen die jij hebt geraadpleegd.

BIJLAGEN

BIJLAGE 1: OSSEC.CONF

```
<!--
  Wazuh - Manager - Default configuration for ubuntu 24.04
  More info at: https://documentation.wazuh.com
  Mailing list: https://groups.google.com/forum/#!forum/wazuh
-->

<ossec_config>
  <global>
    <jsonout_output>yes</jsonout_output>
    <alerts_log>yes</alerts_log>
    <logall>no</logall>
    <logall_json>no</logall_json>
    <email_notification>no</email_notification>
    <smtp_server>smtp.example.wazuh.com</smtp_server>
    <email_from>wazuh@example.wazuh.com</email_from>
    <email_to>recipient@example.wazuh.com</email_to>
    <email_maxperhour>12</email_maxperhour>
    <email_log_source>alerts.log</email_log_source>
    <agents_disconnection_time>10m</agents_disconnection_time>
    <agents_disconnection_alert_time>0</agents_disconnection_alert_time>
    <update_check>yes</update_check>
  </global>

  <alerts>
    <log_alert_level>8</log_alert_level>
    <email_alert_level>12</email_alert_level>
  </alerts>

  <integration>
    <name>custom-shuffle</name>
    <hook_url>https://172.17.0.226:3443/api/v1/hooks/webhook_ebd9e0e1-c840-49af-b05d-384c8e5015be </hook_url> <!-- Replace with your Shuffle hook URL -->
    <level>12</level>
    <alert_format>json</alert_format>
  </integration>

  <!-- Choose between "plain", "json", or "plain,json" for the format of internal logs -->
  <logging>
    <log_format>json</log_format>
  </logging>
```

```
  <remote>
    <connection>secure</connection>
    <port>1514</port>
    <protocol>tcp</protocol>
    <queue_size>131072</queue_size>
  </remote>

  <!-- Policy monitoring -->
  <rootcheck>
    <disabled>no</disabled>
    <check_files>yes</check_files>
    <check_trojans>yes</check_trojans>
    <check_dev>yes</check_dev>
    <check_sys>yes</check_sys>
    <check_pids>yes</check_pids>
    <check_ports>yes</check_ports>
    <check_if>yes</check_if>

    <!-- Frequency that rootcheck is executed - every 12 hours -->
    <frequency>43200</frequency>

    <rootkit_files>etc/rootcheck/rootkit_files.txt</rootkit_files>
    <rootkit_trojans>etc/rootcheck/rootkit_trojans.txt</rootkit_trojans>

    <skip_nfs>yes</skip_nfs>

    <ignore>/var/lib/containerd</ignore>
    <ignore>/var/lib/docker/overlay2</ignore>
  </rootcheck>

  <wodle name="cis-cat">
    <disabled>yes</disabled>
    <timeout>1800</timeout>
    <interval>1d</interval>
    <scan-on-start>yes</scan-on-start>

    <java_path>wodles/java</java_path>
    <ciscat_path>wodles/ciscat</ciscat_path>
  </wodle>
```

```
  <!-- Osquery integration -->
  <wodle name="osquery">
    <disabled>yes</disabled>
    <run_daemon>yes</run_daemon>
    <log_path>/var/log/osquery/osqueryd.results.log</log_path>
    <config_path>/etc/osquery/osquery.conf</config_path>
    <add_labels>yes</add_labels>
  </wodle>

  <!-- System inventory -->
  <wodle name="syscollector">
    <disabled>no</disabled>
    <interval>1h</interval>
    <scan_on_start>yes</scan_on_start>
    <hardware>yes</hardware>
    <os>yes</os>
    <network>yes</network>
    <packages>yes</packages>
    <ports all="no">yes</ports>
    <processes>yes</processes>

    <!-- Database synchronization settings -->
    <synchronization>
      <max_eps>10</max_eps>
    </synchronization>
  </wodle>

  <sca>
    <enabled>yes</enabled>
    <scan_on_start>yes</scan_on_start>
    <interval>12h</interval>
    <skip_nfs>yes</skip_nfs>
  </sca>

  <vulnerability-detection>
    <enabled>yes</enabled>
    <index-status>yes</index-status>
    <feed-update-interval>60m</feed-update-interval>
  </vulnerability-detection>
```

```
  <indexer>
    <enabled>yes</enabled>
    <hosts>
      <host>https://172.17.0.232:9200</host>
    </hosts>
    <ssl>
      <certificate_authorities>
        <ca>/etc/filebeat/certs/root-ca.pem</ca>
      </certificate_authorities>
      <certificate>/etc/filebeat/certs/filebeat.pem</certificate>
      <key>/etc/filebeat/certs/filebeat-key.pem</key>
    </ssl>
  </indexer>

  <!-- File integrity monitoring -->
  <syscheck>
    <disabled>no</disabled>

    <!-- Frequency that syscheck is executed default every 12 hours -->
    <frequency>43200</frequency>

    <scan_on_start>yes</scan_on_start>

    <!-- Generate alert when new file detected -->
    <alert_new_files>yes</alert_new_files>

    <!-- Don't ignore files that change more than 'frequency' times -->
    <auto_ignore frequency="10" timeframe="3600">no</auto_ignore>

    <!-- Directories to check  (perform all possible verifications) -->
    <directories>/etc,/usr/bin,/usr/sbin</directories>
    <directories>/bin,/sbin,/boot</directories>

    <!-- Files/directories to ignore -->
    <ignore>/etc/mtab</ignore>
    <ignore>/etc/hosts.deny</ignore>
    <ignore>/etc/mail/statistics</ignore>
    <ignore>/etc/random-seed</ignore>
    <ignore>/etc/random.seed</ignore>
    <ignore>/etc/adjtime</ignore>
    <ignore>/etc/httpd/logs</ignore>
    <ignore>/etc/utmpx</ignore>
    <ignore>/etc/wtmpx</ignore>
```

```
    <ignore>/etc/cups/certs</ignore>
    <ignore>/etc/dumpdates</ignore>
    <ignore>/etc/svc/volatile</ignore>

    <!-- File types to ignore -->
    <ignore type="sregex">.log$|.swp$</ignore>

    <!-- Check the file, but never compute the diff -->
    <nodiff>/etc/ssl/private.key</nodiff>

    <skip_nfs>yes</skip_nfs>
    <skip_dev>yes</skip_dev>
    <skip_proc>yes</skip_proc>
    <skip_sys>yes</skip_sys>

    <!-- Nice value for Syscheck process -->
    <process_priority>10</process_priority>

    <!-- Maximum output throughput -->
    <max_eps>50</max_eps>

    <!-- Database synchronization settings -->
    <synchronization>
      <enabled>yes</enabled>
      <interval>5m</interval>
      <max_eps>10</max_eps>
    </synchronization>
  </syscheck>

  <!-- Active response -->
  <global>
    <white_list>127.0.0.1</white_list>
    <white_list>^localhost.localdomain$</white_list>
    <white_list>127.0.0.53</white_list>
  </global>

  <command>
    <name>disable-account</name>
    <executable>disable-account</executable>
    <timeout_allowed>yes</timeout_allowed>
  </command>
```

```
  <command>
    <name>restart-wazuh</name>
    <executable>restart-wazuh</executable>
  </command>

  <command>
    <name>firewall-drop</name>
    <executable>firewall-drop</executable>
    <timeout_allowed>yes</timeout_allowed>
  </command>

  <command>
    <name>host-deny</name>
    <executable>host-deny</executable>
    <timeout_allowed>yes</timeout_allowed>
  </command>

  <command>
    <name>route-null</name>
    <executable>route-null</executable>
    <timeout_allowed>yes</timeout_allowed>
  </command>

  <command>
    <name>win_route-null</name>
    <executable>route-null.exe</executable>
    <timeout_allowed>yes</timeout_allowed>
  </command>

  <command>
    <name>netsh</name>
    <executable>netsh.exe</executable>
    <timeout_allowed>yes</timeout_allowed>
  </command>

  <!--
  <active-response>
    active-response options here
  </active-response>
  -->
```

```
  <!-- Log analysis -->
  <localfile>
    <log_format>command</log_format>
    <command>df -P</command>
    <frequency>360</frequency>
  </localfile>

  <localfile>
    <log_format>full_command</log_format>
    <command>netstat -tulpn | sed 's/\([[:alnum:]]\+\)\ \+[[:digit:]]\+\ \+[[:digit:]]\+\ \+\(.*\):\([[:digit:]]*\)\ \+\([0-9\.\:\*]\+\).\+\ \([[:digit:]]*
    <alias>netstat listening ports</alias>
    <frequency>360</frequency>
  </localfile>

  <localfile>
    <log_format>full_command</log_format>
    <command>last -n 20</command>
    <frequency>360</frequency>
  </localfile>

  <ruleset>
    <!-- Default ruleset -->
    <decoder_dir>ruleset/decoders</decoder_dir>
    <rule_dir>ruleset/rules</rule_dir>
    <rule_exclude>0215-policy_rules.xml</rule_exclude>
    <list>etc/lists/audit-keys</list>
    <list>etc/lists/amazon/aws-eventnames</list>
    <list>etc/lists/security-eventchannel</list>

    <!-- User-defined ruleset -->
    <decoder_dir>etc/decoders</decoder_dir>
    <rule_dir>etc/rules</rule_dir>
  </ruleset>

  <rule_test>
    <enabled>yes</enabled>
    <threads>1</threads>
    <max_sessions>64</max_sessions>
    <session_timeout>15m</session_timeout>
  </rule_test>
```

```
  <!-- Configuration for wazuh-authd -->
  <auth>
    <disabled>no</disabled>
    <port>1515</port>
    <use_source_ip>no</use_source_ip>
    <purge>yes</purge>
    <use_password>no</use_password>
    <ciphers>HIGH:!ADH:!EXP:!MD5:!RC4:!3DES:!CAMELLIA:@STRENGTH</ciphers>
    <!-- <ssl_agent_ca></ssl_agent_ca> -->
    <ssl_verify_host>no</ssl_verify_host>
    <ssl_manager_cert>etc/sslmanager.cert</ssl_manager_cert>
    <ssl_manager_key>etc/sslmanager.key</ssl_manager_key>
    <ssl_auto_negotiate>no</ssl_auto_negotiate>
  </auth>

  <cluster>
    <name>wazuh</name>
    <node_name>node01</node_name>
    <node_type>master</node_type>
    <key></key>
    <port>1516</port>
    <bind_addr>0.0.0.0</bind_addr>
    <nodes>
        <node>NODE_IP</node>
    </nodes>
    <hidden>no</hidden>
    <disabled>yes</disabled>
  </cluster>

</ossec_config>
```

Bijlagen vormen een uitstekend middel om de kern van het document leesbaar te houden. Een voorwaarde is wel dat de tekst zonder gebruikmaking van de bijlagen een begrijpelijk geheel vormt. Je mag de lezer nooit dwingen heen en weer te bladeren.
Naar elke bijlage dient minstens een keer verwezen te worden in het rapport.

TIP'S VOOR TEKSTEN IN (DOCUMENTEN OP) JE AFSTUDEERPORTFOLIO

Zie Canvas https://thomasmore.instructure.com/courses/39858/files/8637578?module_item_id=2616078