THOMAS MORE

Netleaf Adversary Emulation

Realisatiedocument

Siebe Moeskops
Student Bachelor in de Toegepaste Informatica – Applicatieontwikkeling
Student Bachelor in de Elektronica-ICT – Cloud & Cyber Security

2025 - 2026

Inhoudsopgave

1. Introductie

Dit realisatiedocument bouwt verder op het project charter, waarin de doelstellingen, planning en scope van het project werden vastgelegd. Waar het project charter beschrijft wat gerealiseerd zou worden, behandelt dit document de effectieve uitvoering van het project en de behaalde resultaten. (Moeskops, 2026)

De focus ligt op de ontwikkeling van een geautomatiseerde oplossing voor detectievalidatie binnen het Security Operations Center (SOC) van Netleaf. Hierbij wordt toegelicht hoe aanvalssimulaties werden ingezet om detectieregels binnen Rapid7 InsightIDR te valideren en hoe deze aanpak resulteerde in een detection coverage mapping, een Detection-as-Code pipeline en een oplossing voor continue detectievalidatie.

Daarnaast worden de analyse, methodologie, implementatie, resultaten en bevindingen van het project besproken. Tot slot wordt gereflecteerd op de technische uitdagingen, leerpunten en mogelijke toekomstige uitbreidingen.

2. Context van de stage

De stage vond plaats binnen Netleaf, een organisatie gespecialiseerd in managed Security Operations Center (SOC)-diensten. Binnen deze context lag de focus op adversary emulation en detectievalidatie binnen een bestaande SOC-omgeving.

De stageopdracht had als doel een geautomatiseerde oplossing te ontwikkelen waarmee aanvalssimulaties kunnen worden gebruikt om detectieregels binnen Rapid7 InsightIDR te valideren. Hierbij werd onderzocht hoe Atomic Red Team-tests gekoppeld kunnen worden aan bestaande detecties om inzicht te verkrijgen in de detectiedekking en mogelijke detection gaps.

De opdracht bestond uit drie onderdelen:

- Het ontwikkelen van een detection coverage mapping waarbij Atomic Red Team-tests gekoppeld worden aan detectieregels binnen Rapid7 InsightIDR.
- Het ontwikkelen van een Detection-as-Code pipeline waarmee nieuwe detectieregels automatisch getest en gevalideerd kunnen worden.
- Het implementeren van continue detectievalidatie door bestaande detectieregels periodiek te testen aan de hand van aanvalssimulaties.

Het uiteindelijke doel was het ontwikkelen van een oplossing die inzicht biedt in de effectiviteit van detectieregels en de kwaliteit van de detectieomgeving op een geautomatiseerde manier kan bewaken.

3. Analyse

Voor de realisatie van het project werd eerst onderzocht welke technologieën, platformen en methodieken het best aansloten bij de projectdoelstellingen. Hierbij lag de focus op het uitvoeren van aanvalssimulaties, het valideren van detecties, het automatiseren van workflows en het visualiseren van resultaten.

Naast de technische mogelijkheden werd ook rekening gehouden met factoren zoals compatibiliteit, schaalbaarheid, gebruiksgemak en integratiemogelijkheden binnen de bestaande SOC-omgeving. Op basis van deze analyse werd een combinatie van bestaande Netleaf-technologieën en aanvullende tools geselecteerd.

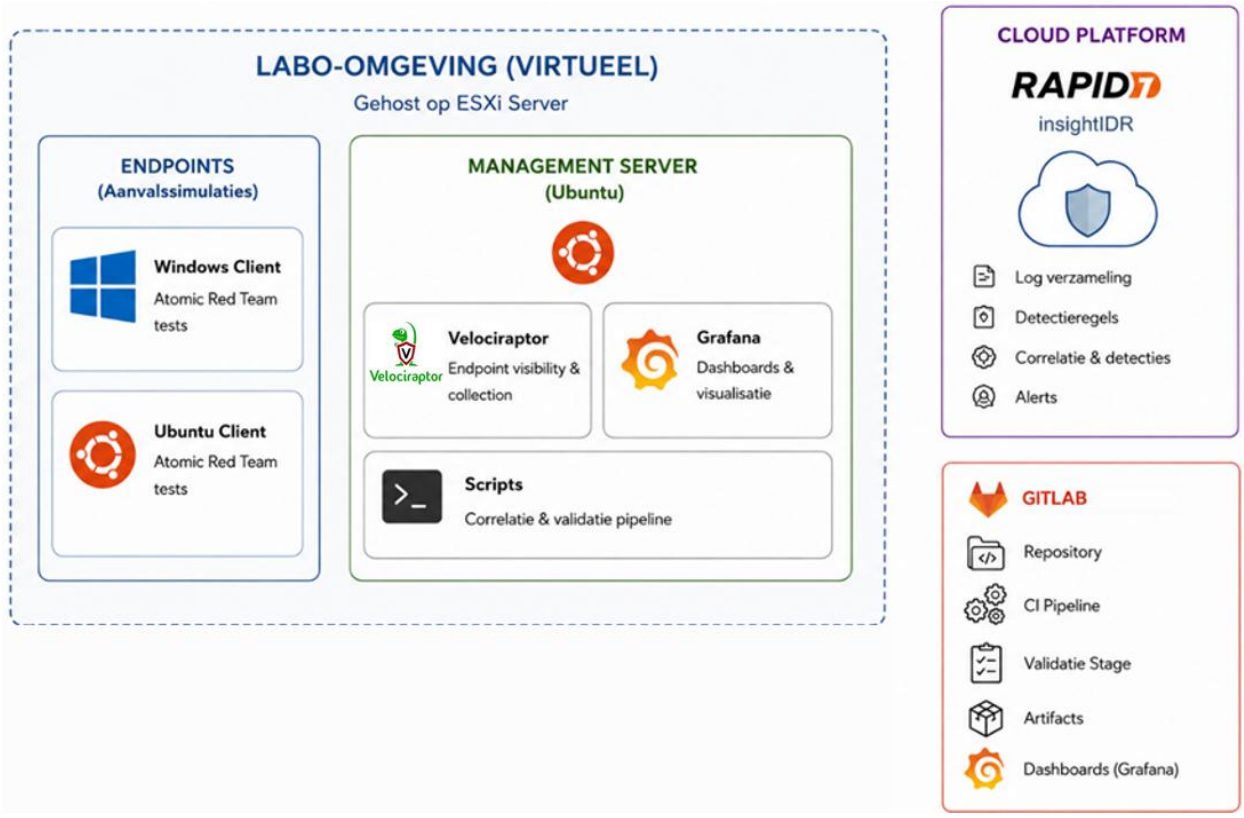
3.1. Labo-omgeving

Voor de realisatie van het project werd een afzonderlijke labo-omgeving opgezet op een VMware ESXi-server. Deze omgeving werd volledig gescheiden gehouden van productieomgevingen zodat aanvalssimulaties veilig konden worden uitgevoerd.

De omgeving bestond uit een centrale Ubuntu-managementserver waarop Velociraptor, Grafana en de validatiescripts werden gehost. Daarnaast werden een Windows- en Ubuntu-endpoint voorzien waarop Atomic Red Team-tests werden uitgevoerd. Deze endpoints hebben ook de installatie van de Atomic Red Team-Tests gehad zodat deze toestellen deze correct kunnen uitvoeren. Zie *figuur 1*.

Voor detectie en loganalyse werd gebruikgemaakt van een afzonderlijke Rapid7 InsightIDR-omgeving. Hierdoor konden detectieregels en aanvalssimulaties getest worden zonder impact op klantomgevingen. GitLab werd gebruikt voor versiebeheer en het automatiseren van pipelines.

Figuur 1 Architectuur van de labo-omgeving



Opmerking. Overzicht van de virtuele labo-omgeving. **Eigen werk.**

3.2. Analyse van de gebruikte tools

3.2.1. Rapid7 InsightIDR

Rapid7 InsightIDR is het centrale SIEM-platform binnen de stageomgeving en vormde daardoor een logische basis voor dit project. Het platform verzamelt logs, genereert alerts en maakt gebruik van detectieregels om verdachte activiteiten te identificeren.

Rapid7 werd gekozen omdat het al operationeel aanwezig was binnen de organisatie en directe toegang bood tot detectieregels, alerts en API-functionaliteiten. Hierdoor kon het platform gebruikt worden als bron voor de validatie van detecties.

3.2.2. Atomic Red Team

Atomic Red Team is een open-source framework dat MITRE ATT&CK-technieken omzet in concrete en reproduceerbare testen. Deze testen simuleren realistische aanvalstechnieken op systemen.

Atomic Red Team werd gekozen omdat het een eenvoudige en gestructureerde manier biedt om specifieke aanvalstechnieken uit te voeren zonder complexe malware of zware infrastructuur. Daarnaast sluit het framework rechtstreeks aan bij MITRE ATT&CK, wat de mapping met detecties vereenvoudigt.

3.2.3. MITRE ATT&CK Framework

MITRE ATT&CK is een internationaal erkend framework dat aanvalstechnieken categoriseert op basis van tactieken en gedrag van aanvallers.

Binnen dit project werd MITRE ATT&CK gebruikt als referentiekader om aanvalstechnieken te structureren en te koppelen aan Atomic tests en detectieregels. Hierdoor ontstaat een gestandaardiseerde methode om detectiedekking te analyseren.

3.2.4. Velociraptor

Velociraptor is een platform voor endpoint monitoring, remote execution en digitale forensische analyse. Velociraptor werd gekozen om Atomic Red Team-tests centraal en geautomatiseerd uit te voeren op verschillende systemen binnen de labo-omgeving. Hierdoor was manuele uitvoering niet langer nodig en konden testen sneller en consistenter worden uitgevoerd.

3.2.5. Grafana

Grafana is een platform voor datavisualisatie en monitoring dashboards. Grafana werd gekozen om de resultaten van de detectievalidatie op een overzichtelijke manier weer te geven. Dashboards maken het mogelijk om PASS/FAIL-resultaten, detectiedekking en trends snel te interpreteren.

3.2.6. GitLab

GitLab werd gebruikt voor versiebeheer en het automatiseren van pipelines via CI/CD-functionaliteiten. GitLab werd gekozen omdat het toelaat om workflows te automatiseren, testen op vaste momenten uit te voeren en scripts centraal te beheren. Hierdoor kon een geautomatiseerde detectievalidatiepipeline worden opgebouwd.

3.3. Weighted Ranking Methode (WRM)

Om de verschillende technologieën objectief te vergelijken, werd gebruikgemaakt van de Weighted Ranking Methode (WRM). Hierbij werden verschillende criteria beoordeeld die essentieel waren voor het behalen van de projectdoelstellingen. zoals automatisatie en integratie met de bestaande omgeving, kregen een hoger gewicht dan minder kritische criteria.

3.3.1. Vergelijking testuitvoering/adversary emulation

Voor de uitvoering van aanvalssimulaties werden verschillende oplossingen onderzocht. Naast manuele uitvoering werden zowel Velociraptor als MITRE Caldera geëvalueerd. Caldera biedt uitgebreide mogelijkheden voor adversary emulation, maar vereist meer configuratie en infrastructuur. Velociraptor sloot beter aan bij de bestaande omgeving en bood voldoende automatisatiemogelijkheden voor de scope van dit project. Zie *tabel 1.*

Tabel 1 WRM Adversary Emulation

Criteria	Gewicht	Velociraptor	Caldera	Manuele uitvoering/scripts
Automatisatie	5	4	4	2
Integratie met bestaande omgeving	5	5	2	3
Gebruiksgemak	4	4	4	3
Schaalbaarheid	3	3	4	1
Implementatiesnelheid	3	3	3	2
Totaalscore		**78**	**61**	**44**

De criteria werden beoordeeld op een schaal van 1 tot 5, waarbij een hogere score een betere aansluiting op de projectdoelstellingen betekent. Voor deze vergelijking werd extra nadruk gelegd op automatisatie en integratie met de bestaande omgeving, aangezien aanvalssimulaties op een centrale en reproduceerbare manier uitgevoerd moesten kunnen worden. Schaalbaarheid en implementatiesnelheid werden als ondersteunende criteria meegenomen.

3.3.2. Vergelijking rapportering

Voor het visualiseren van resultaten werden Grafana, Kibana en Excel onderzocht. Kibana biedt krachtige visualisatiemogelijkheden binnen de Elastic Stack, maar integreerde minder goed met de bestaande omgeving. Excel is eenvoudig in gebruik en geschikt voor beperkte datasets, maar ondersteunt geen geautomatiseerde dashboards of real-time updates. Zie *tabel 2*.

Tabel 2 WRM Visualisatie Tools

Criteria	Gewicht	Grafana	Kibana	Excel
Visualisatie	5	5	4	2
Automatische updates	5	5	5	1
Integratie met bestaande omgeving	5	5	2	2
Integratie databank	4	5	4	3
Gebruiksgemak	3	4	4	5
Totaalscore		**102**	**86**	**52**

Binnen deze vergelijking kregen visualisatie, automatische updates en integratie met de bestaande omgeving het hoogste gewicht. Het was immers belangrijk dat validatieresultaten centraal, overzichtelijk en automatisch konden worden weergegeven zonder extra manuele verwerking.

3.3.3. Vergelijking pipeline automatisatie

Voor de automatisatie van workflows werden GitLab CI/CD, Jenkins en Cronjobs onderzocht. Jenkins is een krachtig automatiseringsplatform met uitgebreide uitbreidingsmogelijkheden, maar vereist meer configuratie en beheer. Cronjobs zijn eenvoudig te implementeren voor geplande taken, maar missen centrale logging, versiebeheer en geavanceerde workflow functionaliteiten. Zie *tabel 3.*

Tabel 3 WRM Pipeline Tools

Criteria	Gewicht	GitLab CI/CD	Jenkins	Cronjobs
Automatisatie	5	5	5	3
Integratie met bestaande omgeving	4	5	3	4
Schaalbaarheid	4	5	5	3
Gebruiksgemak	3	4	3	5
Centrale beheerbaarheid	4	5	4	3
Totaalscore		**97**	**84**	**66**

Voor pipeline-automatisatie lag de focus voornamelijk op automatisatie, schaalbaarheid en centrale beheerbaarheid. Aangezien de oplossing verschillende validatieprocessen automatisch moest uitvoeren, waren betrouwbaarheid en eenvoudige integratie binnen de bestaande ontwikkelomgeving van groot belang.

3.4. Verantwoording van de keuzes

Op basis van de uitgevoerde vergelijkingen werd gekozen voor een combinatie van Rapid7 InsightIDR, Atomic Red Team, Velociraptor, GitLab CI/CD en Grafana. Naast de technische mogelijkheden speelde ook de integratie met de bestaande infrastructuur een belangrijke rol bij deze keuzes.

Rapid7 InsightIDR vormde al het centrale SIEM-platform binnen het SOC van Netleaf en bood de nodige API-functionaliteiten voor het ophalen en verwerken van detecties. Voor het uitvoeren van aanvalssimulaties werd gekozen voor Atomic Red Team in combinatie met Velociraptor. Deze combinatie maakte het mogelijk om reproduceerbare testen centraal en geautomatiseerd uit te voeren zonder complexe bijkomende infrastructuur.

Voor pipeline-automatisatie werd gekozen voor GitLab CI/CD. GitLab werd reeds gebruikt binnen de organisatie voor versiebeheer en automatisatie, waardoor de ontwikkelde oplossing eenvoudig geïntegreerd kon worden in bestaande workflows. Voor rapportering en visualisatie werd Grafana geselecteerd, aangezien dit platform al aanwezig was binnen de labo-omgeving en eenvoudig gekoppeld kon worden aan de databank met validatieresultaten.

Door maximaal gebruik te maken van technologieën die reeds aanwezig waren binnen de bestaande SOC-omgeving, kon de oplossing sneller worden gerealiseerd en bleef de integratie met bestaande processen beperkt. Tegelijk werden open-source tools toegevoegd waar deze een duidelijke meerwaarde boden voor detectievalidatie en rapportering.

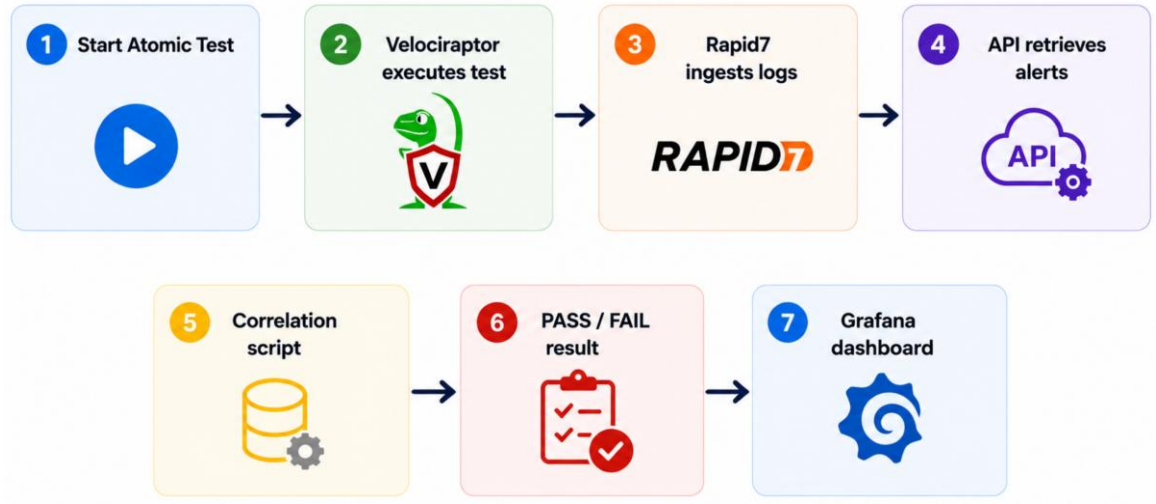
4. Methodologie

In dit hoofdstuk wordt de werking van de ontwikkelde oplossing stap voor stap toegelicht. Om detectieregels automatisch te kunnen valideren, werd een workflow ontwikkeld die aanvalssimulaties uitvoert, detecties verzamelt, resultaten correleert en deze vervolgens visualiseert in een centraal dashboard.

Zoals weergegeven in *figuur 2* bestaat deze workflow uit zeven opeenvolgende stappen. Elke stap vervult een specifieke functie binnen het validatieproces en draagt bij aan het automatisch bepalen van de effectiviteit van detectieregels binnen Rapid7 InsightIDR.

De volgende secties beschrijven de werking van elke stap afzonderlijk, van het uitvoeren van een Atomic Red Team-test tot het visualiseren van de validatieresultaten in Grafana.

Figuur 2 Methodologie Flowdiagram



Opmerking. Overzicht van de workflow. **Eigen werk.**

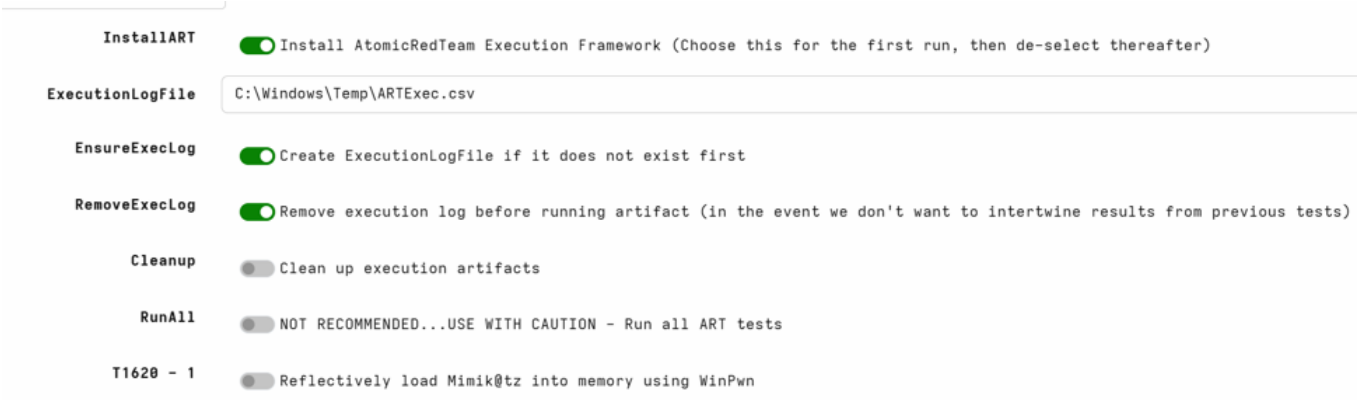
4.1. Start van Atomic Red-Team test

Het validatieproces start met het selecteren van één of meerdere Atomic Red Team-tests die uitgevoerd moeten worden. Deze selectie gebeurt op basis van de aanvalstechnieken die getest of gevalideerd moeten worden. Elke Atomic test is gekoppeld aan een specifieke techniek en bevat de commando's die nodig zijn om bepaald aanvalsgedrag te simuleren op een endpoint.

Binnen de oplossing worden de gekozen testen opgenomen in een testbatch. Een batch bevat dus één of meerdere testen die samen worden uitgevoerd binnen hetzelfde validatiemoment. Hierdoor kan later duidelijk bepaald worden welke testen op welk moment gestart zijn en welke detecties binnen datzelfde tijdsvenster verwacht worden.

De geselecteerde testen worden niet rechtstreeks manueel uitgevoerd op het endpoint. In plaats daarvan wordt de uitvoering centraal aangestuurd via Velociraptor. Dit zorgt ervoor dat de testuitvoering reproduceerbaar blijft en dat dezelfde testen op een gecontroleerde manier opnieuw uitgevoerd kunnen worden. Zie *figuur 3*.

Figuur 3 Selecteer Atomic Red Team-Tests



Opmerking. Overzicht van de selectie om Atomic Red Team-Tests uit te voeren. **Velociraptor**

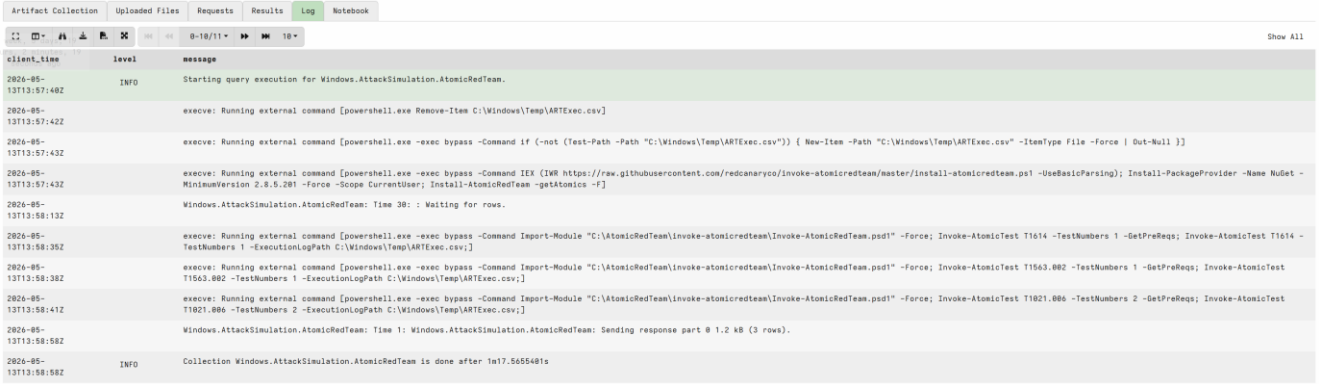
4.2. Uitvoering van Velociraptor

Velociraptor wordt gebruikt als centrale component om de Atomic Red Team-tests uit te voeren op het gekozen endpoint. Vanuit de managementserver wordt een Velociraptor artifact gestart dat de geselecteerde Atomic tests uitvoert op het target systeem.

Tijdens deze stap wordt bepaald op welk endpoint de test moet plaatsvinden, bijvoorbeeld op een Windows- of Ubuntu-client. Vervolgens voert Velociraptor de bijhorende commando's uit op dat systeem. Deze commando's veroorzaken gecontroleerde systeemactiviteiten die overeenkomen met het gedrag dat door de Atomic test wordt gesimuleerd. Zie *figuur 4*.

Na de uitvoering registreert Velociraptor belangrijke uitvoeringsinformatie. Hierbij gaat het onder andere om de uitgevoerde test, de gebruikte techniek, het doelendpoint, het tijdstip van uitvoering en de status van de uitvoering. Deze gegevens zijn belangrijk omdat ze later gebruikt worden om de gegenereerde detecties correct te koppelen aan de juiste test.

Figuur 4 Uitvoering Atomic Red Team-Tests



Opmerking. Voorbeeld van een uitvoering van Atomic Red Team-Tests. **Velociraptor.**

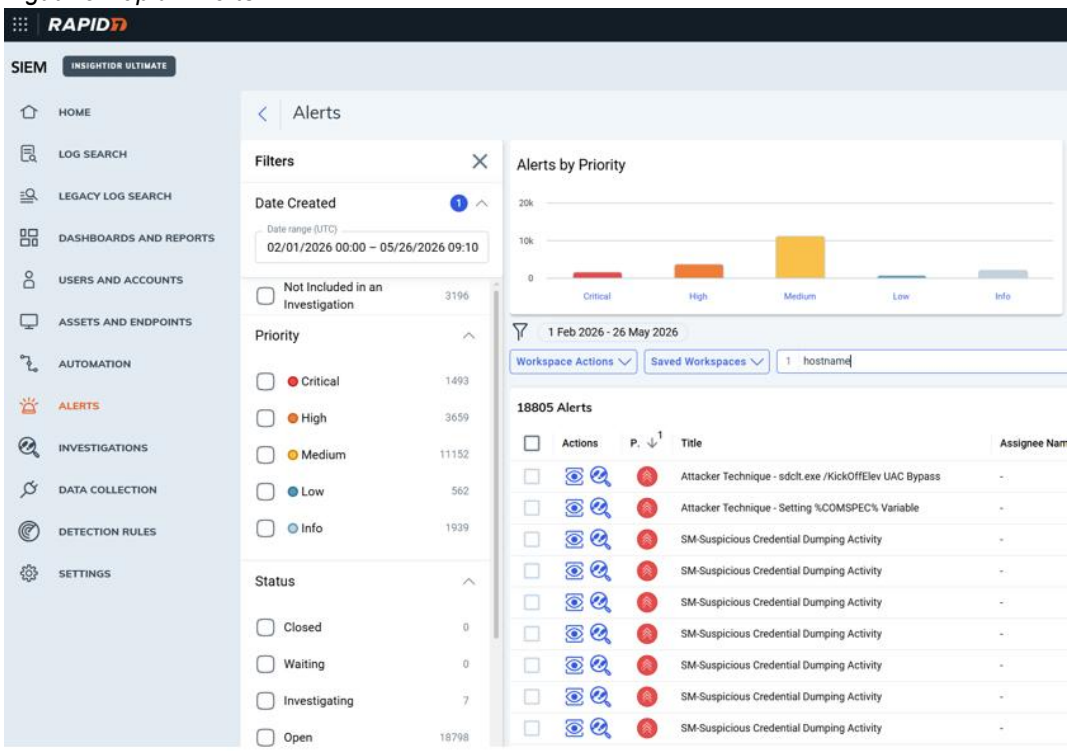
4.3. Rapid7 InsightIDR log ingestie

Wanneer de Atomic test wordt uitgevoerd op het endpoint, ontstaan er systeemactiviteiten zoals procesuitvoering, command execution, file activity of andere logging events. Deze gebeurtenissen worden verzameld door de Rapid7-agent die actief is op het endpoint, zoals weergeven in *figuur 5*.

De verzamelde logs worden doorgestuurd naar Rapid7 InsightIDR. Binnen Rapid7 worden deze logs verwerkt en gecontroleerd tegenover de aanwezige detectieregels. Wanneer het gedrag overeenkomt met een bestaande detectieregel, genereert Rapid7 een detectie of alert.

Deze stap is belangrijk omdat de kwaliteit van de validatie afhankelijk is van correcte logverzameling. Als een endpoint geen juiste logs doorstuurt of als de logverwerking vertraging heeft, kan een detectieregel mogelijk niet correct gevalideerd worden. Daarom wordt er in de workflow rekening gehouden met een wachttijd na de uitvoering van de test, zodat Rapid7 voldoende tijd krijgt om de logs te verwerken.

Figuur 5 Rapid7 Alerts



Opmerking. Overzicht van de alerts in Rapid7. **Rapid7**

4.4. Alerts ophalen via de Rapid7 API

Na de uitvoering van de testen en de ingestie van logs worden de nodige gegevens opgehaald voor verdere verwerking. De pipeline haalt hiervoor informatie op uit twee bronnen. Enerzijds worden de uitvoeringsgegevens uit Velociraptor gebruikt. Deze informatie toont welke Atomic tests effectief uitgevoerd zijn, op welk endpoint dit gebeurde en binnen welk tijdsvenster de uitvoering plaatsvond.

Anderzijds worden via de Rapid7 API de gegenereerde alerts opgehaald. Hierbij wordt gezocht naar alerts die binnen het ingestelde validatievenster zijn aangemaakt. Dit validatievenster start rond het moment waarop de Atomic tests werden uitgevoerd en houdt rekening met mogelijke vertraging in logverwerking.

Door de gegevens uit Velociraptor en Rapid7 samen te brengen, beschikt de oplossing over twee datasets: de uitgevoerde aanvalssimulaties en de detecties die daar mogelijk uit voortkwamen. Deze datasets vormen de input voor het correlatiescript.

4.5. Correlatie script

Het correlatiescript vormt de kern van de validatieflow. Dit script vergelijkt de uitgevoerde Atomic tests met de opgehaalde alerts uit Rapid7 InsightIDR. Het doel is om te bepalen of een specifieke test ook effectief geleid heeft tot een relevante detectie.

Eerst verwerkt het script de resultaten uit Velociraptor. Daarbij wordt bepaald welke testen uitgevoerd werden, welke techniek getest werd en op welk endpoint dit gebeurde. Vervolgens verwerkt het script de alerts die via de Rapid7 API werden opgehaald.

Daarna past het script verschillende filters toe om ruis te beperken. Eerst wordt gekeken naar het endpoint waarop de test werd uitgevoerd. Alerts van andere systemen worden uitgesloten, omdat die niet relevant zijn voor de validatie van deze specifieke test. Daarna wordt gecontroleerd of de alert binnen het ingestelde tijdsvenster valt. Alerts buiten dit venster worden niet meegenomen, omdat de kans groter is dat ze niet veroorzaakt werden door de uitgevoerde test.

Vervolgens wordt gekeken naar welke detectie regels er opgehaald zijn door de API. Op die manier kan het script bepalen of de welke alert er in de batchtijd zijn binnengekomen, en dit matchen met de aanvalstechniek die door de Atomic test werd uitgevoerd.

Door deze combinatie van hostfiltering, tijdsvensters en techniek- of regelmapping kan het script bepalen welke alerts relevant zijn voor een specifieke test. Het resultaat van deze stap is een correlatie tussen uitgevoerde testen en gegenereerde detecties. Zie *figuur 6.*

Figuur 6 Output Correlatie Script

```
🚀 Batch started
🕒 START : 2026-04-20 12:27:07
🕒 END   : 2026-04-20 12:32:07
⏳ WAIT  : 300 sec (5 min)
🧪 Requested Atomic Tests:
   → T1021.006:2
   → T1563.002:1
   → T1614:1
🕒 Actual END : 2026-04-20 12:32:07
🔍 Checking 1 latest flow
📁 FLOW: /opt/velociraptor/clients/C.b23f9f501c895923/collections/F.D7J1N3VJHP446
🧪 Executed Atomic Tests:
   → Technique: T1021.006 | Test: 2
   → Technique: T1563.002 | Test: 1
   → Technique: T1614 | Test: 1
🚨 Alerts fetched: 19
📊 Detection summary:
   → STANDARD: 4
   → CUSTOM  : 8
🎯 RESULT: PASS
✅ Detection validation PASSED
Cleaning up project directory and file based variables
Job succeeded
```

Opmerking. Overzicht van de output van het correlatie script. **Eigen Werk**

4.6. Validatie van het resultaat

Na de correlatie wordt voor elke uitgevoerde test een validatiestatus bepaald. Deze status geeft aan of de verwachte detectie effectief werd teruggevonden.

Tijdens de eerste fase van het project lag de focus voornamelijk op het in kaart brengen van de bestaande detectiedekking binnen Rapid7 InsightIDR. Hierbij werd niet alleen gekeken naar zelf ontwikkelde detectieregels, maar ook naar de standaard detectieregels die al aanwezig waren binnen het platform. Het doel van deze fase was om een zo volledig mogelijk beeld te krijgen van welke aanvalstechnieken reeds werden gedetecteerd en waar mogelijke detection gaps aanwezig waren. Wanneer een relevante detectie werd teruggevonden, ongeacht of deze afkomstig was van een standaardregel of een custom regel, werd de test als succesvol beschouwd.

Naarmate het project verder evolueerde, veranderde ook de validatielogica. Binnen de Detection-as-Code pipeline werd de focus verlegd naar het testen van nieuw ontwikkelde detectieregels. In deze situatie volstond het niet langer om eender welke detectie te ontvangen. De pipeline moest specifiek controleren of de nieuw aangemaakte detectieregel correct werd geactiveerd. Hierdoor werd de validatie strenger en werd enkel gekeken naar de verwachte custom detectieregel die door de pipeline werd uitgerold.
Wanneer de verwachte detectieregel werd teruggevonden binnen het ingestelde tijdsvenster, kreeg de test de status PASS. Als de verwachte regel niet werd geactiveerd, werd een FAIL-status toegekend. Op deze manier kon automatisch bepaald worden of een nieuwe detectieregel correct functioneerde alvorens deze verder gebruikt werd binnen de detectieomgeving.

De validatieresultaten worden opgeslagen zodat ze later gebruikt kunnen worden voor rapportering, trendanalyse en continue opvolging van de detectiekwaliteit.

4.7. Visualisatie in Grafana

De resultaten van de correlatie en validatie worden centraal opgeslagen in een databank. Grafana leest deze gegevens uit en zet ze om in dashboards die de status van de detectievalidatie overzichtelijk weergeven.

De dashboards tonen onder andere welke testen uitgevoerd werden, welke testen een PASS- of FAIL-status kregen, welke detectieregels geactiveerd werden en waar mogelijke detection gaps aanwezig zijn. Daarnaast kunnen trends zichtbaar worden, bijvoorbeeld wanneer een detectieregel eerder wel werkte maar bij latere validaties niet meer correct triggert.

Door deze visualisatie krijgt het SOC-team een centraal overzicht van de actuele detectiedekking. Dit maakt het eenvoudiger om problemen in detectieregels op te volgen, verouderde regels te identificeren en de algemene kwaliteit van de detectieomgeving te bewaken.

5. Implementatie van de oplossing

Na de analyse van de beschikbare technologieën en het uitwerken van de methodologie, werd de overgang ingezet naar de effectieve implementatie van de oplossing. De implementatie werd opgebouwd rond drie centrale doelstellingen die samen een volledig validatieproces voor detectieregels moesten vormen. Hiervoor koppelen we even terug naar de voorop samengestelde doelstellingen.

De eerste doelstelling bestond uit het ontwikkelen van een detection coverage mapping. Voor Netleaf was het op dat moment niet zichtbaar welke detectieregels effectief reageerden op specifieke aanvalstechnieken en waar er nog detection gaps aanwezig waren. Daarom werd eerst onderzocht hoe Atomic Red Team-tests gekoppeld konden worden aan detecties binnen Rapid7 InsightIDR. Het doel van deze fase was het creëren van een betrouwbaar overzicht van de bestaande detectiedekking.

Zodra een betrouwbare correlatiemethode beschikbaar was, werd deze uitgebreid naar een Detection-as-Code pipeline. Het doel van deze tweede fase was het automatiseren van het validatieproces voor nieuwe detectieregels. Hierdoor konden nieuw ontwikkelde regels automatisch worden uitgerold, getest en gevalideerd alvorens ze verder gebruikt werden binnen de detectieomgeving. Op deze manier kon de kwaliteit van nieuwe detecties op een consistente en reproduceerbare manier gecontroleerd worden.

De derde doelstelling richtte zich op het continu monitoren van bestaande detectieregels. Hoewel een detectieregel correct kan functioneren op het moment dat deze ontwikkeld wordt, bestaat het risico dat wijzigingen in logbronnen, detectielogica of infrastructuur ervoor zorgen dat deze na verloop van tijd niet meer correct werkt. Daarom werd een oplossing ontwikkeld waarmee bestaande detectieregels periodiek opnieuw gevalideerd kunnen worden. Het doel van deze fase was het identificeren van verouderde of niet-functionerende detecties en het bewaken van de algemene gezondheid van de detectieomgeving.

De volgende secties beschrijven hoe deze drie doelstellingen tijdens de stage werden gerealiseerd en welke technische oplossingen hiervoor werden ontwikkeld.

5.1. Task 1 - Coverage Mapping

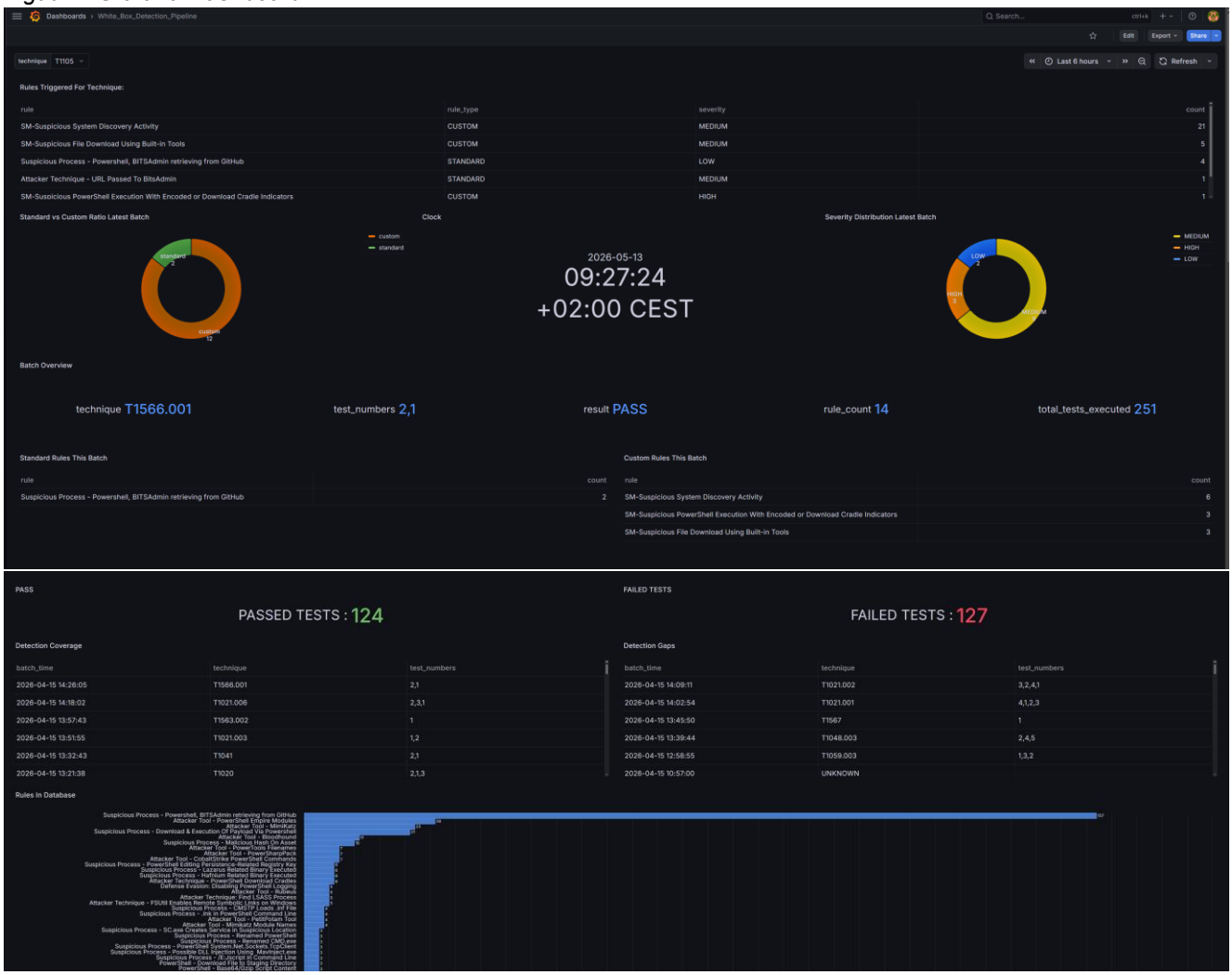
De eerste fase van het project bestond uit het ontwikkelen van een detection coverage mapping. Het doel hiervan was om inzicht te verkrijgen in welke aanvalstechnieken reeds werden gedetecteerd door de bestaande detectieregels binnen Rapid7 InsightIDR.

Hiervoor werden verschillende Atomic Red Team-tests geselecteerd voor zowel Windows- als Linux-systemen. Deze testen werden uitgevoerd binnen de labo-omgeving, waarna de gegenereerde detecties werden geanalyseerd. In eerste instantie gebeurde deze analyse manueel om inzicht te krijgen in de werking van de detectieregels en de gegenereerde alerts.

Om dit proces schaalbaar te maken werd een eigen correlatiescript ontwikkeld. Dit script koppelde de uitgevoerde Atomic tests aan de gegenereerde detecties op basis van hostinformatie, tijdsvensters en MITRE ATT&CK-technieken. Hierdoor kon automatisch bepaald worden welke technieken succesvol werden gedetecteerd en waar mogelijke detection gaps aanwezig waren.

De resultaten werden opgeslagen in een centrale databank en gevisualiseerd via Grafana. Hierdoor ontstond een overzicht van de actuele detectiedekking binnen de omgeving en werd zichtbaar welke technieken al ondersteund werden door standaard- of custom detectieregels.

Figuur 7 Grafana Dashboard 1



Opmerking. Overzicht van het eerste Grafana dashboard. **Grafana**

Deze eerste fase resulteerde in een detection coverage mapping waarbij voor elke uitgevoerde Atomic Red Team-test werd bepaald welke detectieregels binnen Rapid7 InsightIDR hierop reageerden. Op basis van deze analyse werd voor iedere test een koppeling opgebouwd tussen de aanvalssimulatie en de bijhorende detectieregels.

Binnen het dashboard zoals weergeven in *figuur 7* wordt onderscheid gemaakt tussen testen waarvoor een standaard Rapid7-detectieregel beschikbaar is en testen die uitsluitend door custom detectieregels worden gedetecteerd. De groene resultaten vertegenwoordigen technieken waarvoor een standaardregel werd teruggevonden, terwijl de rode resultaten technieken weergeven waarvoor enkel een custom detectieregel beschikbaar is.

Daarnaast werd tijdens deze fase een centrale databank opgebouwd waarin alle validatieresultaten worden opgeslagen. Hierdoor kan voor elke Atomic Red Team-test een historiek van de uitgevoerde validaties worden bijgehouden en gevisualiseerd via de dashboards.

5.2. Task 2 - Detection-as-Code Pipeline

Nadat een betrouwbare correlatiemethode ontwikkeld was, werd deze uitgebreid naar een volledig geautomatiseerde Detection-as-Code pipeline. Het doel van deze fase was om het volledige proces van het aanmaken, testen en valideren van nieuwe detectieregels te automatiseren.

Voor deze oplossing werd een aparte GitLab-omgeving opgezet waarin alle validatiestappen centraal beheerd worden. In plaats van configuraties hardcoded in scripts op te nemen, werd gekozen om alle belangrijke parameters onder te brengen in GitLab CI/CD-variabelen. Hierdoor kunnen nieuwe validaties eenvoudig worden gestart door enkel de gewenste Atomic test, target host of validatie-instellingen aan te passen zonder wijzigingen aan de broncode.

De uitvoering van de pipeline gebeurt via een GitLab Runner die gehost wordt op een Ubuntu-server binnen de labo-omgeving. Deze runner vormt de centrale uitvoeringsomgeving voor alle pipelinestappen en beschikt over toegang tot Rapid7 InsightIDR, Velociraptor, Terraform en de benodigde validatiescripts. Hierdoor kunnen alle onderdelen van de oplossing vanuit één centrale locatie worden aangestuurd.

Voor het beheer van detectieregels werd gebruikgemaakt van Terraform. Nieuwe detectieregels worden gedefinieerd als code en automatisch uitgerold naar Rapid7 InsightIDR tijdens de pipeline-uitvoering. Op deze manier wordt de volledige configuratie versie beheerbaar en reproduceerbaar gemaakt. Iedere wijziging aan een detectieregel wordt bijgehouden binnen GitLab, waardoor wijzigingen eenvoudig opgevolgd en teruggedraaid kunnen worden. Zie *figuur 8.*

Figuur 8 Terraform output

```
$ terraform apply -parallelism=1 -auto-approve
rapid7_siem_detection_rule.rules["powershell_test"]: Refreshing state... [name=SM-Test PowerShell]
rapid7_siem_detection_rule.rules["siebe_rule_3"]: Refreshing state... [name=Siebe rule 3]
Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  + create
Terraform will perform the following actions:
  # rapid7_siem_detection_rule.rules["bash_test"] will be created
  + resource "rapid7_siem_detection_rule" "rules" {
      + action      = "CREATES_ALERTS"
      + description = "Test rule via Terraform"
      + logic       = {
          + leql = "from(event_type = \"process_start_event\") where(process.name = \"bash\")"
        }
      + name        = "SM-Test Bash"
      + priority    = "INFO"
      + rrn         = (known after apply)
      + version_rrn = (known after apply)
    }
  # rapid7_siem_detection_rule.rules["powershell_atomic_smoke_test"] will be created
  + resource "rapid7_siem_detection_rule" "rules" {
      + action      = "CREATES_ALERTS"
      + description = "Smoke test rule for validating Atomic Red Team PowerShell execution"
      + logic       = {
          + leql = "from(event_type = \"process_start_event\") where(process.name = \"powershell.exe\")"
        }
      + name        = "SM-Test PowerShell Atomic Smoke"
      + priority    = "INFO"
      + rrn         = (known after apply)
      + version_rrn = (known after apply)
    }
Plan: 2 to add, 0 to change, 0 to destroy.
Changes to Outputs:
  ~ rule_names = {
      + powershell_atomic_smoke_test = "SM-Test PowerShell Atomic Smoke"
        # (3 unchanged attributes hidden)
    }
  ~ rule_rrns  = {
      ~ bash_test                  = "rrn:cba:eu:[MASKED]:custom-rule:IVHKSRDL1ICS" -> (known after apply)
      + powershell_atomic_smoke_test = (known after apply)
        # (2 unchanged attributes hidden)
    }
rapid7_siem_detection_rule.rules["bash_test"]: Creating...
rapid7_siem_detection_rule.rules["bash_test"]: Creation complete after 0s [name=SM-Test Bash]
rapid7_siem_detection_rule.rules["powershell_atomic_smoke_test"]: Creating...
rapid7_siem_detection_rule.rules["powershell_atomic_smoke_test"]: Creation complete after 1s [name=SM-Test PowerShell Atomic Smoke]
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Opmerking. Overzicht van het aanmaken van nieuwe regels met Terraform. **Terraform**

Na het uitrollen van de nieuwe regels start de pipeline automatisch de bijhorende Atomic Red Team-test via Velociraptor, zoals de methodologie eerder vermeld. Vervolgens worden de gegenereerde alerts opgehaald uit Rapid7 InsightIDR en verwerkt door het validatiescript. Hierbij wordt gecontroleerd of de nieuw aangemaakte detectieregel effectief werd geactiveerd door de uitgevoerde aanvalssimulatie.

Een belangrijk onderdeel van de implementatie was het onderscheid maken tussen bestaande en nieuw aangemaakte detectieregels. In eerdere versies van de pipeline werden alle aanwezige regels gevalideerd, wat leidde tot onnodige validaties van regels die al succesvol getest waren. Daarom werd een mechanisme ontwikkeld dat enkel de regels valideert die tijdens de huidige pipeline-run werden aangemaakt of aangepast. Hierdoor blijft de validatie beperkt tot de wijzigingen die daadwerkelijk getest moeten worden.

De validatie gebeurt op basis van de detectieregels die tijdens de huidige pipeline-run werden aangemaakt of gewijzigd. Nadat de Atomic Red Team-tests zijn uitgevoerd, wordt gedurende een vooraf bepaald validatievenster gecontroleerd of deze nieuwe regels effectief een alert genereren binnen Rapid7 InsightIDR.

Wanneer een regel minstens één keer succesvol wordt geactiveerd door de uitgevoerde aanvalssimulatie, wordt deze als gevalideerd beschouwd. De regel blijft vervolgens aanwezig binnen Rapid7 InsightIDR en wordt behouden in de Terraform state voor toekomstig gebruik.

Indien een regel tijdens het volledige validatievenster geen enkele keer wordt geactiveerd, wordt in dat geval de regel beschouwd als niet-functioneel of onvoldoende gevalideerd. De pipeline verwijdert vervolgens enkel deze gefaalde regels automatisch uit Rapid7 InsightIDR, terwijl succesvol gevalideerde regels behouden blijven. Hierdoor kunnen ontwikkelaars zich bij een volgende validatieronde uitsluitend focussen op de regels die nog verdere aanpassingen vereisen. Zie *figuur 9*.

Figuur 9 GitLab Detection-as-Code Pipeline

Opmerking. Overzicht van de Detection-as-Code Pipeline. **GitLab**

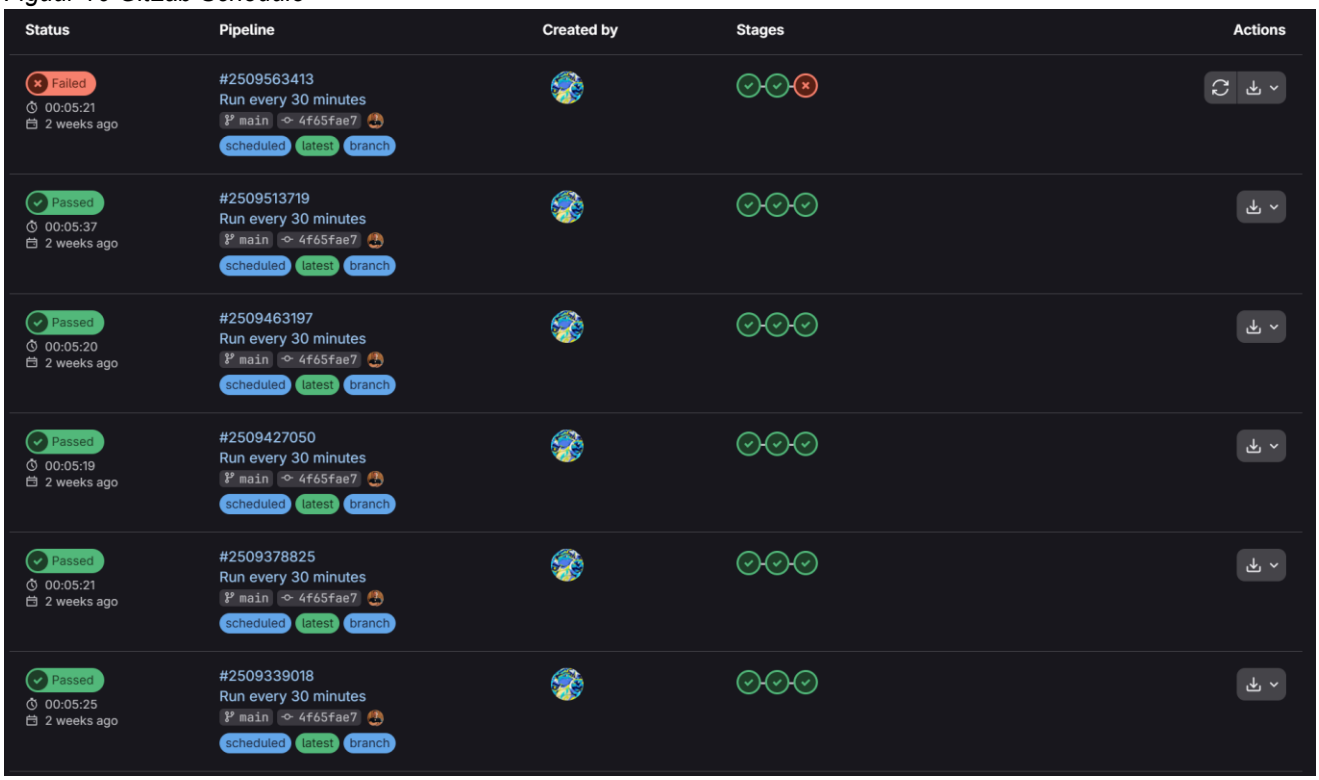
Hierdoor blijven werkende regels behouden terwijl enkel de foutieve regels opnieuw ontwikkeld en getest moeten worden. Deze aanpak zorgde ervoor dat nieuwe detectieregels op een gecontroleerde, reproduceerbare en schaalbare manier ontwikkeld konden worden. Bovendien werd het manuele proces van aanmaken, testen, controleren en verwijderen van detectieregels volledig geautomatiseerd.

5.3. Task 3 - Continue Detectie Validatie

Na de implementatie van de Detection-as-Code pipeline werd een derde oplossing ontwikkeld voor de continue validatie van bestaande detectieregels. Waar de vorige pipeline zich richtte op het testen van nieuw ontwikkelde detectieregels, lag de focus hier op het periodiek controleren van de algemene gezondheid van de detectieomgeving.

Om het laatste probleem aan te pakken werd een afzonderlijke GitLab-pipeline ontwikkeld die automatisch wordt uitgevoerd via een geplande taak. De pipeline wordt iedere 30 minuten gestart zoals in *figuur 10* en selecteert telkens een beperkte batch van Atomic Red Team-tests uit een vooraf gedefinieerde testset. Hierdoor wordt de omgeving continu blootgesteld aan gecontroleerde aanvalssimulaties zonder de infrastructuur onnodig zwaar te belasten.

Figuur 10 GitLab Schedule



Opmerking. Overzicht van het schedule dat GitLab runt. **GitLab**

De oplossing bestaat uit drie opeenvolgende fasen. In de eerste fase wordt een selectie gemaakt van drie tot vijf Atomic tests die tijdens de huidige validatieronde uitgevoerd zullen worden. Hierbij wordt rekening gehouden met vooraf uitgesloten testen, zoals testen die mogelijks het systeem kunnen uitschakelen te excluderen waardoor de pipeline anders zou falen. De geselecteerde testen worden opgeslagen zodat later exact kan worden nagegaan welke technieken verwacht werden binnen deze validatiebatch.

Vervolgens voert Velociraptor de geselecteerde testen uit op het doelendpoint. Tijdens deze uitvoering wordt een unieke flow aangemaakt binnen Velociraptor. Deze flow bevat alle uitvoeringsinformatie die later gebruikt wordt tijdens de validatiefase. Hierdoor kan niet alleen gecontroleerd worden welke testen gevraagd werden, maar ook welke testen daadwerkelijk uitgevoerd zijn.

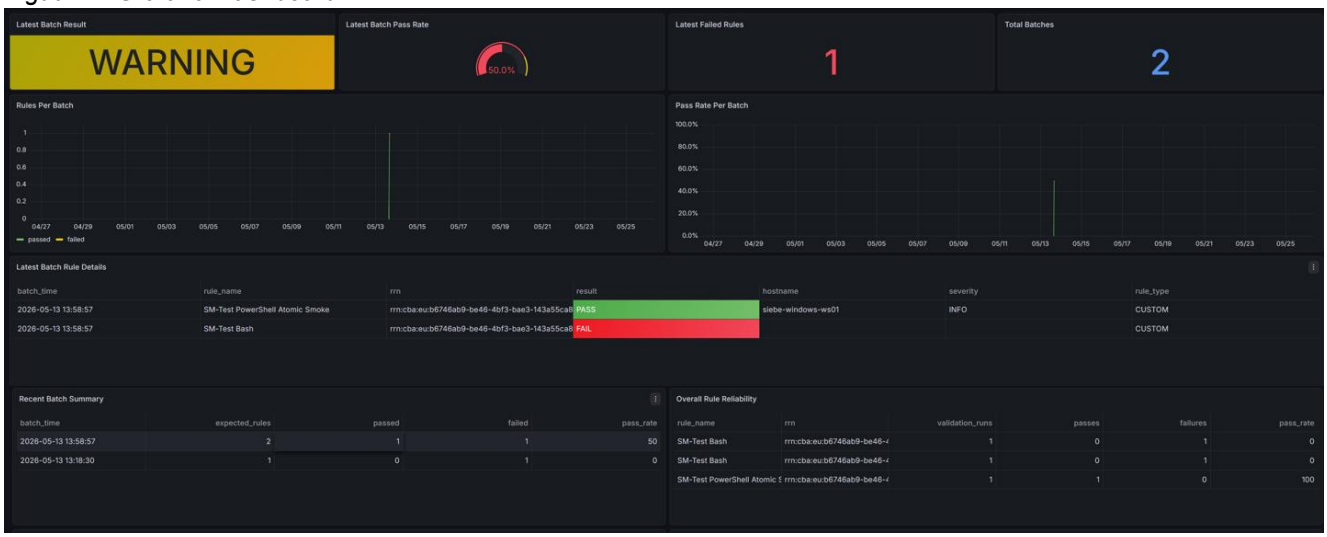
Na een ingestelde wachttijd start de validatiefase. In tegenstelling tot de Detection-as-Code pipeline ligt de focus hier niet op het valideren van een specifieke detectieregel. De oplossing controleert eerst of de gevraagde Atomic tests effectief uitgevoerd werden door de Velociraptor flow logs uit te lezen. Hierdoor kunnen we garanderen dat de testen effectief zijn uitgevoerd. Indien de test niet wordt teruggevonden wordt een FAIL-status toegekend. Op deze manier wordt de betrouwbaarheid van de testuitvoering zelf gecontroleerd.

Naast de validatie van de testuitvoering worden ook alle Rapid7-alerts opgehaald die binnen hetzelfde validatievenster gegenereerd werden. Deze alerts worden niet rechtstreeks gebruikt voor de PASS/FAIL-bepaling, maar geven wel inzicht in welke standaard- en custom detectieregels tijdens de batch geactiveerd werden. Hierdoor ontstaat een continu overzicht van de actuele detectieactiviteit binnen de omgeving.

Alle resultaten worden opgeslagen in een SQLite-databank die gekoppeld is aan Grafana. Na iedere validatie wordt een nieuwe momentopname van de omgeving opgeslagen. Hierdoor ontstaat een historiek van uitgevoerde testen, succesvolle uitvoeringen en geactiveerde detecties. Op basis van deze gegevens kunnen dashboards opgebouwd worden die trends zichtbaar maken en helpen bij het identificeren van mogelijke detection gaps of verouderde detectieregels.

Door deze continue validatie ontstaat een geautomatiseerd controlesysteem dat permanent inzicht geeft in de werking van de detectieomgeving. Hierdoor kunnen afwijkingen sneller worden opgespoord en krijgt het SOC-team een beter overzicht van de kwaliteit en betrouwbaarheid van de aanwezige detectieregels. Zie *figuur 11.*

Figuur 11 Grafana Dashboard 2



Opmerking. Overzicht van het tweede Grafana dashboard. **Grafana**

6. Resultaten en bevindingen

Na de implementatie van de oplossing konden verschillende testen, validaties en analyses worden uitgevoerd. Deze resultaten boden niet alleen inzicht in de effectiviteit van bestaande detectieregels, maar maakten het ook mogelijk om zwakke punten en opportuniteiten binnen de detectieomgeving te identificeren.

In dit hoofdstuk worden de belangrijkste bevindingen besproken. Eerst wordt toegelicht welke inzichten werden verkregen met betrekking tot de detectiedekking. Vervolgens worden de voornaamste beperkingen en knelpunten behandeld die tijdens het project werden vastgesteld.

6.1. Behaalde doelstellingen

De drie vooropgestelde doelstellingen van het project werden succesvol gerealiseerd.

Voor de eerste doelstelling werd een detection coverage mapping ontwikkeld waarmee uitgevoerde Atomic Red Team-tests automatisch gekoppeld kunnen worden aan detectieregels binnen Rapid7 InsightIDR. Door de ontwikkelde correlatielogica werd het mogelijk om voor iedere uitgevoerde test inzicht te verkrijgen in de aanwezige standaard- en custom detectieregels. De resultaten werden centraal opgeslagen in een databank en gevisualiseerd via Grafana-dashboards, waardoor detection gaps en detectiedekking overzichtelijk in kaart gebracht konden worden.

De tweede doelstelling werd gerealiseerd door de ontwikkeling van een volledig geautomatiseerde Detection-as-Code pipeline. Via GitLab CI/CD en Terraform kunnen nieuwe detectieregels automatisch worden uitgerold naar Rapid7 InsightIDR, waarna de bijhorende aanvalssimulaties automatisch worden uitgevoerd. De pipeline valideert vervolgens of de nieuw aangemaakte detectieregels effectief geactiveerd worden. Werkende regels blijven behouden, terwijl foutieve regels automatisch worden verwijderd. Hierdoor werd het volledige proces van ontwikkelen, testen en valideren van detectieregels geautomatiseerd.

Voor de derde doelstelling werd een afzonderlijke oplossing ontwikkeld voor continue detectievalidatie. Via een geplande GitLab-pipeline worden periodiek Atomic Red Team-tests uitgevoerd en gevalideerd. De resultaten worden opgeslagen in een centrale databank en weergegeven via dashboards. Hierdoor ontstaat een historisch overzicht van uitgevoerde testen en geactiveerde detecties, waardoor afwijkingen, mogelijke detection gaps en verouderde detectieregels sneller geïdentificeerd kunnen worden.

De gerealiseerde oplossing biedt hierdoor niet alleen inzicht in de bestaande detectiedekking, maar ondersteunt ook het ontwikkelen van nieuwe detectieregels en het continu bewaken van de kwaliteit van de detectieomgeving binnen het SOC.

6.2. Bevindingen

Tijdens het project werden verschillende waardevolle inzichten verkregen met betrekking tot de werking van detectieregels binnen een SOC-omgeving.

Een eerste belangrijke bevinding was dat de ontwikkelde detection coverage mapping effectief inzicht biedt in de aanwezige detectiedekking. Door Atomic Red Team-tests systematisch uit te voeren en te koppelen aan detecties binnen Rapid7 InsightIDR werd zichtbaar welke aanvalstechnieken al gedetecteerd worden en waar nog detection gaps aanwezig zijn.

Daarnaast werd vastgesteld dat geautomatiseerde validatie een aanzienlijke meerwaarde biedt ten opzichte van manuele detectietesten. Door gebruik te maken van GitLab, Velociraptor en Rapid7 konden validaties op een consistente en reproduceerbare manier worden uitgevoerd. Hierdoor werd de tijd die nodig is om detectieregels te testen aanzienlijk verminderd.

De Detection-as-Code pipeline toonde bovendien aan dat nieuwe detectieregels op een veilige en gecontroleerde manier ontwikkeld kunnen worden. Door regels automatisch uit te rollen, te testen en enkel succesvolle regels te behouden, wordt de kwaliteit van nieuwe detecties verhoogd en neemt de kans op foutieve detectiecontent af.

Ook de continue validatie van bestaande detectieregels leverde waardevolle inzichten op. Door periodiek aanvalssimulaties uit te voeren ontstaat een historisch overzicht van de detectieomgeving. Hierdoor kunnen afwijkingen sneller worden opgemerkt en wordt het eenvoudiger om de kwaliteit van detectieregels op lange termijn te bewaken.

Tot slot bevestigde het project dat een combinatie van aanvalssimulatie, automatisatie en centrale rapportering een effectieve aanpak vormt om detectieomgevingen objectief te evalueren en continu te verbeteren.

6.3. Beperkingen en knelpunten

Tijdens de uitvoering van het project werden ook verschillende technische uitdagingen vastgesteld.

Een eerste uitdaging was het betrouwbaar correleren van aanvalssimulaties met de gegenereerde detecties. Wanneer meerdere testen binnen eenzelfde periode uitgevoerd werden, konden verschillende alerts tegelijkertijd ontstaan. Hierdoor was bijkomende correlatielogica noodzakelijk om de juiste detecties aan de juiste testen te koppelen.

Daarnaast werd vastgesteld dat logverwerking en data-ingestie niet altijd onmiddellijk plaatsvinden. Tussen het uitvoeren van een Atomic test en het verschijnen van een detectie binnen Rapid7 InsightIDR kan enige vertraging optreden. Hierdoor moest rekening gehouden worden met validatievensters en wachttijden om foutieve resultaten te vermijden.

Een bijkomende beperking was dat niet alle Atomic Red Team-tests zonder aanpassingen bruikbaar waren binnen de labo-omgeving. Sommige testen vereisten bijkomende configuratie, specifieke systeeminstellingen of extra softwarecomponenten voordat ze succesvol uitgevoerd konden worden.

Ook het beheer van detectieregels bracht uitdagingen met zich mee. Tijdens de ontwikkeling van de Detection-as-Code pipeline moest een methode worden ontwikkeld om onderscheid te maken tussen al gevalideerde detectieregels en nieuw aangemaakte regels. Zonder deze filtering zouden bestaande regels onnodig opnieuw gevalideerd worden.

Tot slot blijft de oplossing afhankelijk van de correcte werking van verschillende externe componenten, waaronder Velociraptor, Rapid7 InsightIDR, GitLab en de onderliggende endpoint telemetrie. Problemen binnen één van deze componenten kunnen een directe impact hebben op de betrouwbaarheid van de validatieresultaten.

7. Besluit

Tijdens deze stage werd succesvol een geautomatiseerde oplossing ontwikkeld voor het valideren en monitoren van detectieregels binnen een Security Operations Center. Door de combinatie van Atomic Red Team, Velociraptor, Rapid7 InsightIDR, GitLab CI/CD en Grafana werd een platform gerealiseerd waarmee detectiedekking inzichtelijk gemaakt, nieuwe detectieregels automatisch gevalideerd en bestaande detecties continu gecontroleerd kunnen worden.

De gerealiseerde oplossing biedt het SOC-team een objectieve en schaalbare manier om de effectiviteit van detectieregels op te volgen en mogelijke detection gaps sneller te identificeren. Daarnaast toont het project aan dat automatisatie een belangrijke rol kan spelen in het verhogen van de kwaliteit, betrouwbaarheid en onderhoudbaarheid van moderne detectieomgevingen.

Met de ontwikkelde detection coverage mapping, Detection-as-Code pipeline en continue validatie werd een solide basis gelegd voor verdere uitbreiding en optimalisatie van detectievalidatie binnen Netleaf.

8. Glossary

Term	Uitleg
SOC (Security Operations Center)	Team dat beveiligingsincidenten monitort en analyseert.
SIEM (Security Information and Event Management)	Platform voor het verzamelen en analyseren van logs.
Rapid7 InsightIDR	Het SIEM-platform dat binnen Netleaf wordt gebruikt.
Atomic Red Team	Framework voor het simuleren van aanvalstechnieken.
MITRE ATT&CK Framework	Kennisbank die aanvalstechnieken categoriseert.
Velociraptor	Een geavanceerd, open-source platform voor endpoint bewaking, digitale forensische analyse en incidentrespons (DFIR).
Detectieregel	Regel die verdachte activiteiten detecteert.
Correlatie	Het koppelen van een aanvalssimulatie aan een detectie.
Detectiedekking	Overzicht van welke aanvalstechnieken worden gedetecteerd.
Detection gap	Ontbrekende of onvoldoende detectie voor een aanvalstechniek.
Detection-as-Code	Aanpak waarbij detectieregels via code worden beheerd en gevalideerd.
GitLab CI/CD	Functionaliteit voor het automatisch uitvoeren van pipelines.
Adversary Emulation	Het simuleren van aanvallersgedrag om detecties te testen.

LITERATUURLIJST

Center for Threat-Informed Defense. (z.d.). *Atomic Red Team*. MITRE Engenuity. https://atomicredteam.io
GitLab Inc. (z.d.). *GitLab CI/CD documentation*. https://docs.gitlab.com/ee/ci/
Grafana Labs. (z.d.). *Grafana documentation*. https://grafana.com/docs/
HashiCorp. (z.d.). *Terraform documentation*. https://developer.hashicorp.com/terraform/docs
MITRE Corporation. (z.d.). *MITRE ATT&CK® framework*. https://attack.mitre.org
Rapid7. (z.d.). *Alerts*. https://docs.rapid7.com/insightidr/alerts/
Rapid7. (z.d.). *Create and manage basic detection rules*. https://docs.rapid7.com/insightidr/create-and-manage-basic-detection-rules/
Rapid7. (z.d.). *Custom detection rules*. https://docs.rapid7.com/insightidr/custom-detection-rules/
Rapid7. (z.d.). *Detection rules*. https://docs.rapid7.com/insightidr/detection-rules/
Rapid7. (z.d.). *Insight Agent overview*. https://docs.rapid7.com/insight-agent/overview/
Rapid7. (z.d.). *InsightIDR REST API*. https://docs.rapid7.com/insightidr/insightidr-rest-api/
Rapid7. (z.d.). *Managing platform API keys*. https://docs.rapid7.com/insight/managing-platform-api-keys/
Velocidex Enterprises. (z.d.). *Velociraptor API documentation*. https://docs.velociraptor.app/docs/server_automation/server_api/
Velocidex Enterprises. (z.d.). *Velociraptor artifact exchange*. https://docs.velociraptor.app/exchange/
Velocidex Enterprises. (z.d.). *Velociraptor client deployment*. https://docs.velociraptor.app/docs/deployment/clients/
Velocidex Enterprises. (z.d.). *Velociraptor collect_client VQL reference*. https://docs.velociraptor.app/vql_reference/server/collect_client/
Velocidex Enterprises. (z.d.). *Velociraptor flow_logs VQL reference*. https://docs.velociraptor.app/vql_reference/server/flow_logs/
Velocidex Enterprises. (z.d.). *Velociraptor quickstart guide*. https://docs.velociraptor.app/docs/deployment/quickstart/
Velocidex Enterprises. (z.d.). *Windows.AttackSimulation.AtomicRedTeam artifact documentation*. https://docs.velociraptor.app
Moeskops, S. (2026). *Moeskops_Siebe_Project_Charter*.

Bijlagen