The SOC CASE

Ward Boeckx

R0896010

Inhoudstabel

Inleiding

Dit document beschrijft een gedetailleerde opstelling van een Security Operations Center (SOC) door Ward Boeckx. Het SOC is opgebouwd met behulp van verschillende tools en technologieën zoals pfSense, Wazuh, Ossec, en Shuffle, georganiseerd in een virtuele omgeving met specifieke subnetwerken voor productie, aanvalssimulaties, en beveiligingsmonitoring. De documentatie bevat een overzicht van de netwerkconfiguratie, de implementatie van beveiligingstools, en de automatisering van incidentrespons via scripts en API-integraties.

Netwerkopstelling

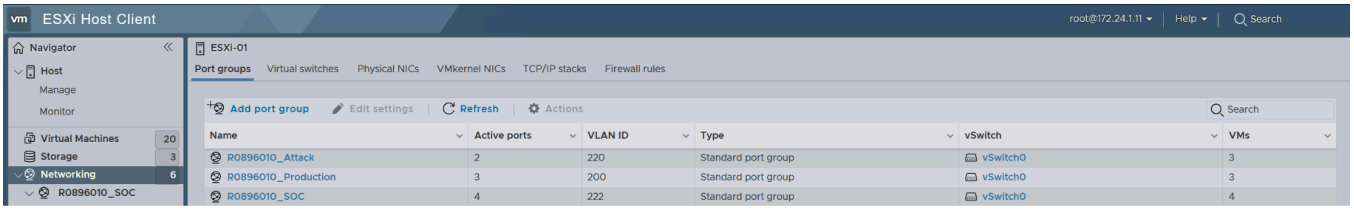
Voor het opzetten van de SOC-omgeving heb ik gebruik gemaakt van een virtuele machine (VM) met pfSense als firewall/router. Deze pfSense-instance is geconfigureerd met vier netwerkinterfaces, die elk een specifiek subnet vertegenwoordigen binnen het netwerk. De interfaces en bijbehorende IP-adressen zijn als volgt geconfigureerd:

- **WAN**: Deze interface is gekoppeld aan het externe netwerk en heeft het IP-adres 172.24.1.107, verkregen via DHCP. Dit subnet fungeert als toegangspunt voor het netwerk van buitenaf.
- **PRODUCTION:**: Het interne subnet voor productie, met het netwerkadres 192.168.1.0/24. Deze interface is bestemd voor de machines in productie, dus machines die geconfigureerd zijn als “agent” en gemonitord worden.
- **ATTACK**: Het interne subnet voor aanvalspogingen, met het netwerkadres 192.168.20.0/24. Deze interface is specifiek bedoeld voor simulaties van aanvallen en andere testen die een afgeschermde omgeving vereisen.
- **SOC**: Het subnet voor het Security Operations Center, met het netwerkadres 192.168.30.0/24. Deze interface dient voor het netwerkverkeer dat verband houdt met het monitoring en beheer van de beveiliging, inclusief de communicatie met de SIEM- en SOAR-systemen.

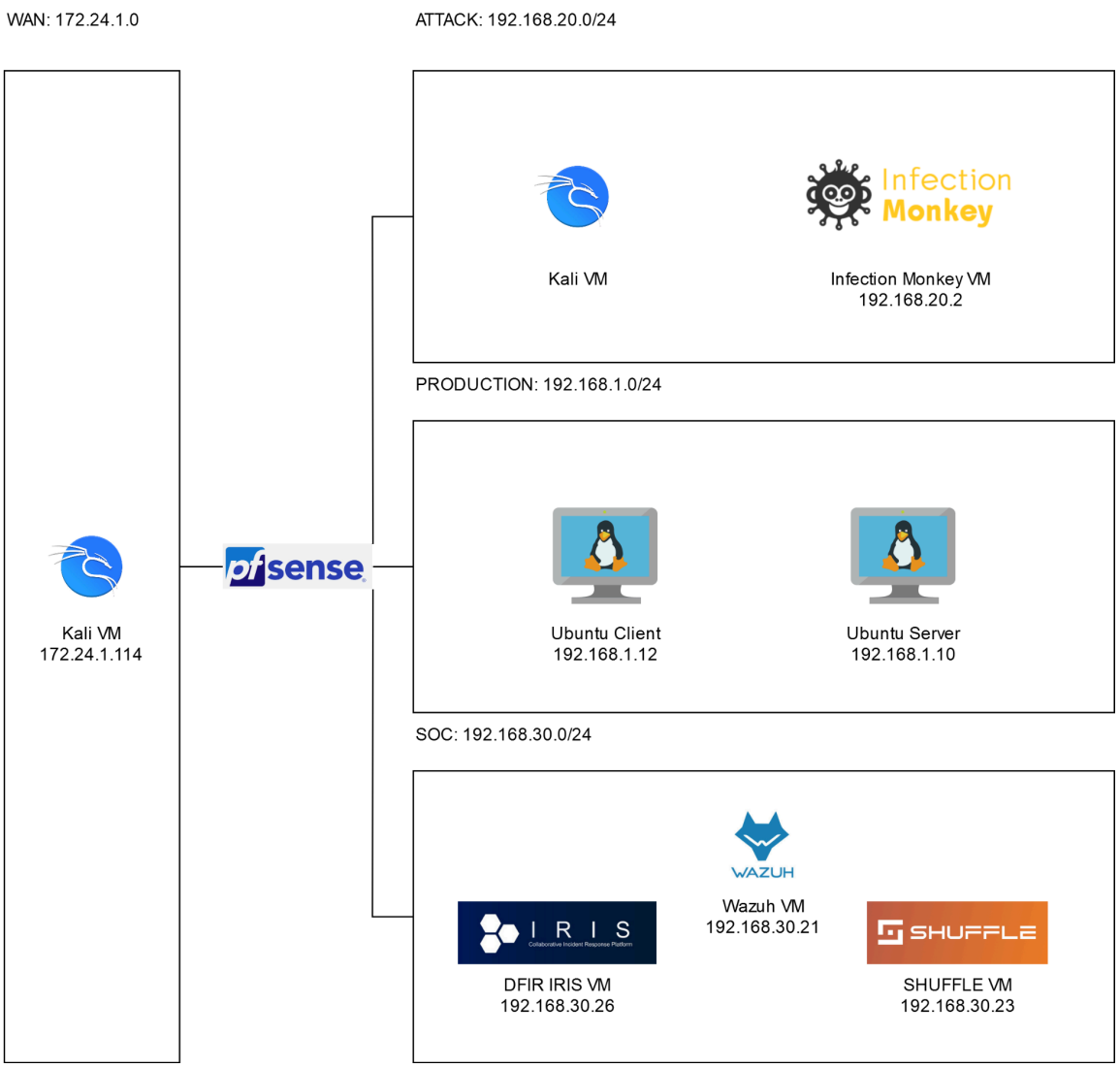
Dit leek mij persoonlijk een logische en goede basis voor het bouwen van de SOC.

```
        groups: pfsync
[2.7.2-RELEASE][root@pfSense_SOC_r0896010.home.arpa]/root:
[2.7.2-RELEASE][root@pfSense_SOC_r0896010.home.arpa]/root: exit
exit
pfSense - Netgate Device ID: 509e539e69f23d38ef8f

*** Welcome to pfSense 2.7.2-RELEASE (amd64) on pfSense_SOC_r0896010 ***

 WAN (wan)       -> vmx0       -> v4/DHCP4: 172.24.1.107/24
 PRODUCTION (lan) -> vmx2       -> v4: 192.168.1.1/24
 ATTACK (opt1)   -> vmx1       -> v4: 192.168.20.1/24
 SOC (opt2)      -> vmx3       -> v4: 192.168.30.1/24
```

Overzicht van de interfaces (PfSense)



Portgroups die zijn toegevoegd op de ESXI Host Client



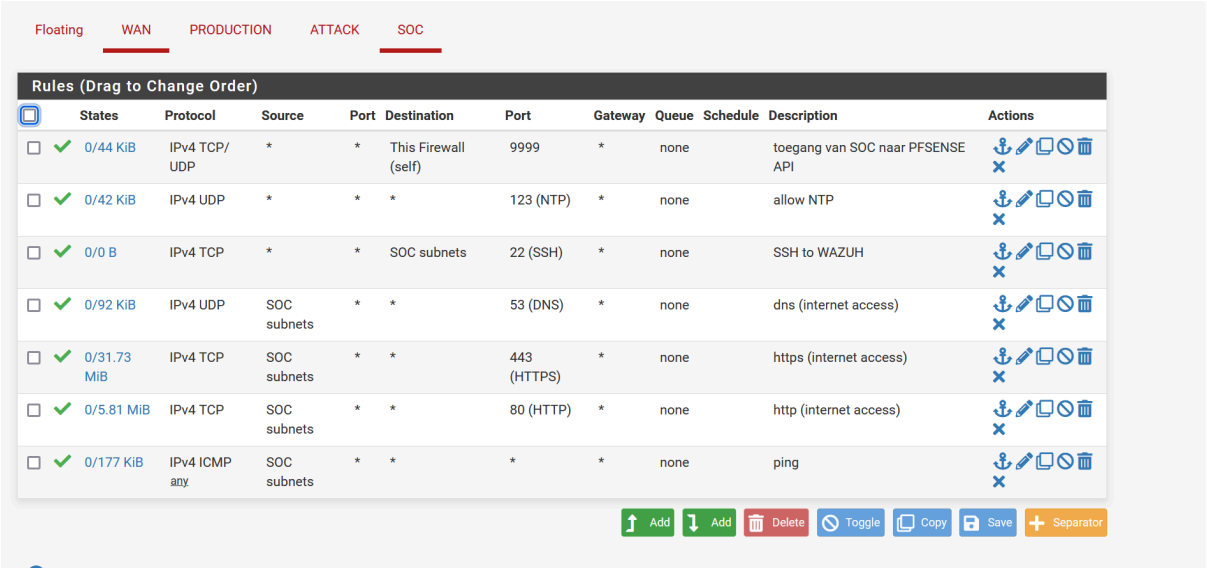
Network overview met VM's

Pfsense

Dieper in op de PfSense instellingen. Deze is redelijk basic buiten de FW Rules, DHCP Server en Port forwarding regels.

De DHCP server binnen het SOC subnet bevat een deel statische IP adressen en dan een DHCP pool die start vanaf **192.168.30.22.** Reden hiervan is omdat het mij logischer lijkt om statische IP adressen te gebruiken voor bepaalde SOC tools. (Wazuh maakt gebruik van integraties die niet meer functioneren als een machine die deel uitmaakt van deze integratie een nieuw IP-adres krijgt.) Waarom dan nog wel een DHCP pool gebruiken? Soms is het handig om snel een machine op te starten om bepaalde dingen te testen en deze een ip te geven via de DHCP server.

Firewall regels



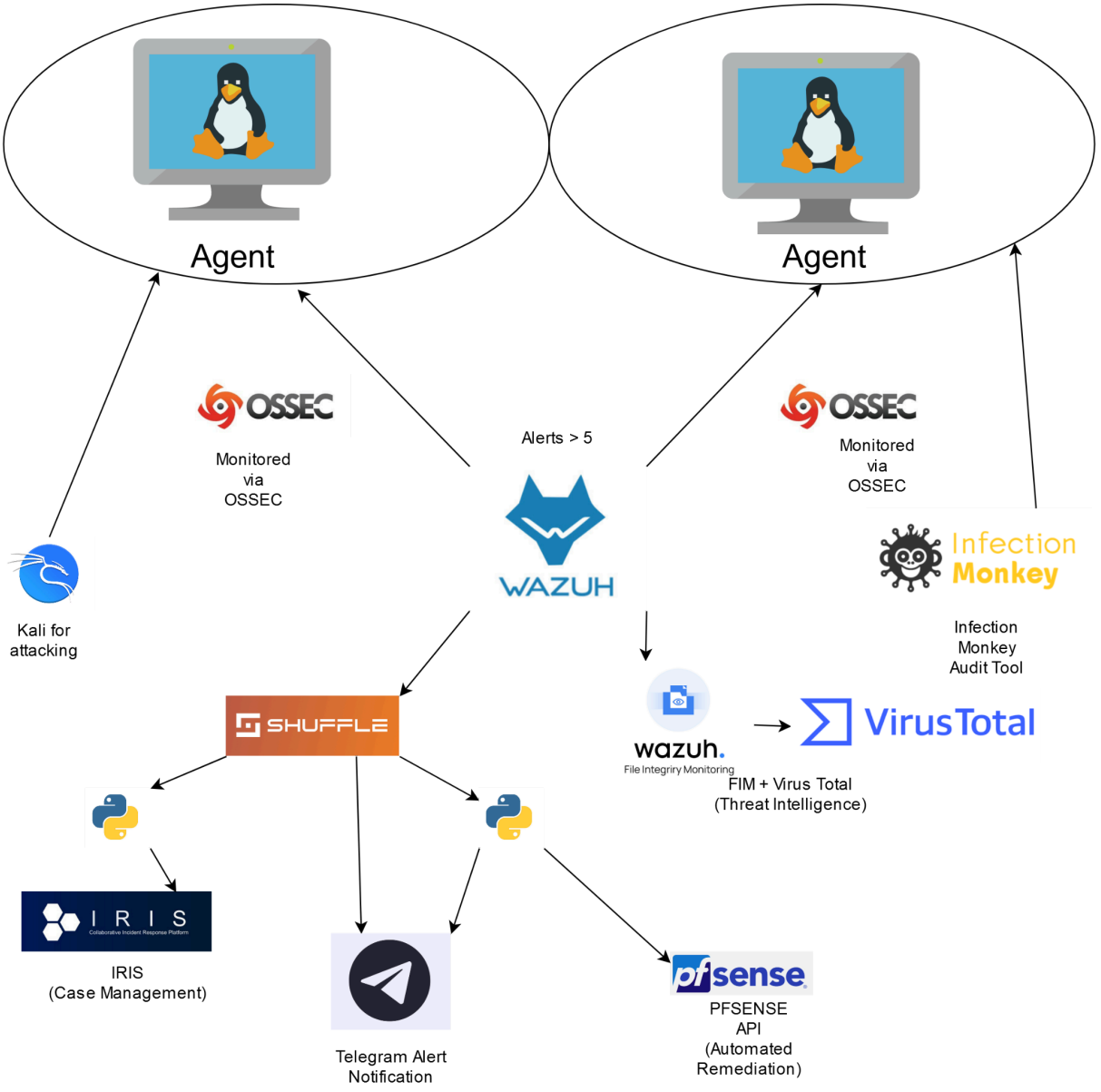
Geprobeerd om op een gecontroleerde manier machines toegang te geven tot het internet en andere protocollen zonder de deuren wijd open te zetten.

Nat / Port Forwarding

Ik gebruik ook Port forwarding van meerdere SOC tools die een Webconfig gebruiken alsook de PfSense Webconfig. Dit is in mijn opinie niet best practice en zou ik in de praktijk ook niet doen, weliswaar nu wel gedaan omwille van demonstratie redenen. Het is gemakkelijk om gewoon even alles te benaderen van mijn eigen laptop zonder te moeten switchen tussen VM's die in bepaalde netwerken zitten.

Soc tools en workflow (birdview)

- **Wazuh - SIEM**
- **Ossec - HIDS**
- **Shuffle, Telegram, PFsense API - Automation Soar**
- **IRIS - DFIR**
- **Virus total - Threat intell.**
- **Infection Monkey - Audit**



Soc workflow.

- De Wazuh-server, die zich in het **SOC-subnet** bevindt, monitort twee virtuele machines in het **Production-subnet.**:
    - **Ubuntu Client (Agent) - 192.168.1.12**

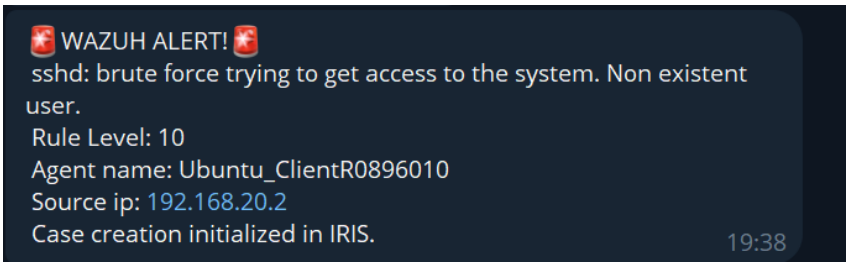
    - **Ubuntu Server (Agent) - 192.168.1.10**
- Deze monitoring gebeurt via **OSSEC**, dat gegevens verzamelt en analyseert om beveiligings alerts te genereren.

Alert verwerking:

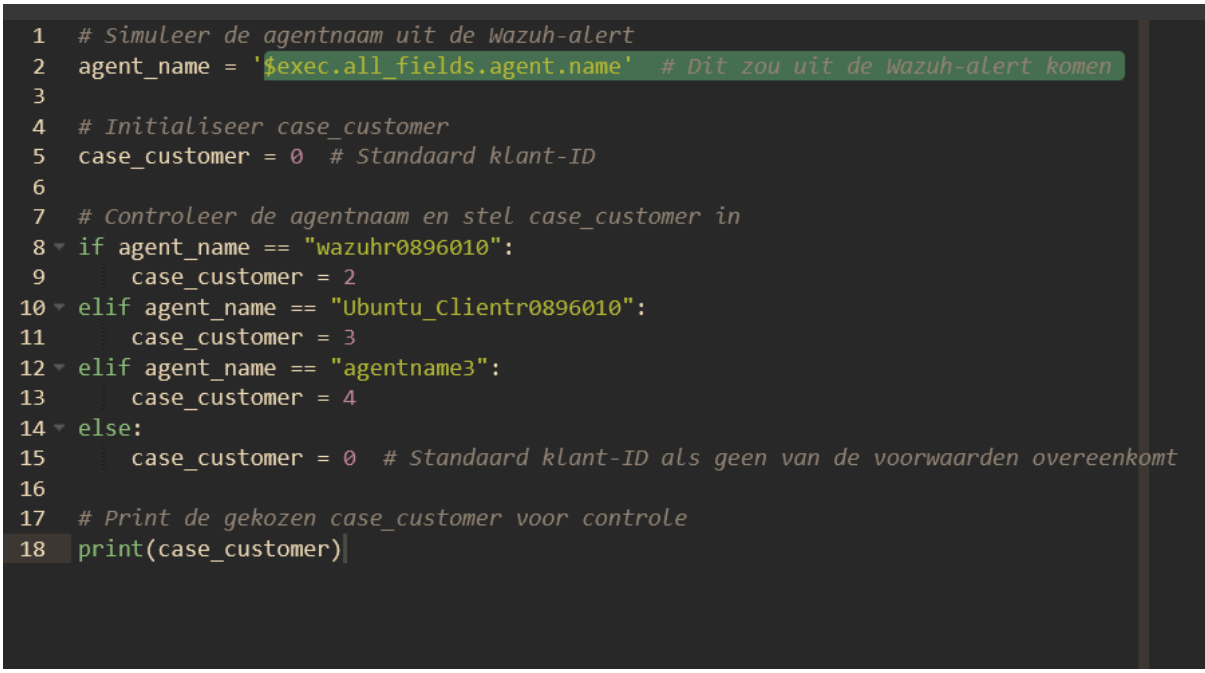
- Wanneer een alert binnenkomt op de Wazuh-server, wordt het niveau van de alert geëvalueerd.
- Alerts met een **prioriteit hoger dan 5** worden doorgestuurd naar **Shuffle.**

Automatisering via Shuffle:

- Shuffle ontvangt de alerts en voert de volgende acties uit:
    - **Telegram-notificatie**: Een bericht met de alert die gefiltered is, wordt automatisch verstuurd naar een Telegram-bot om de SOC-analist direct op de hoogte te brengen van het incident.



    - **Case aanmaken in IRIS**: Voor elke ontvangen alert wordt er automatisch een case aangemaakt in **IRIS** voor verdere analyse en opvolging. Binnen IRIS zitten “Customers”. In deze case gebruik ik de Agents als “Customers”. Alerts geven een “Agent name” mee aan de hand van die naam wordt er dmv een Python script de juiste "customer" meegegeven aan het de Case creation binnen IRIS.



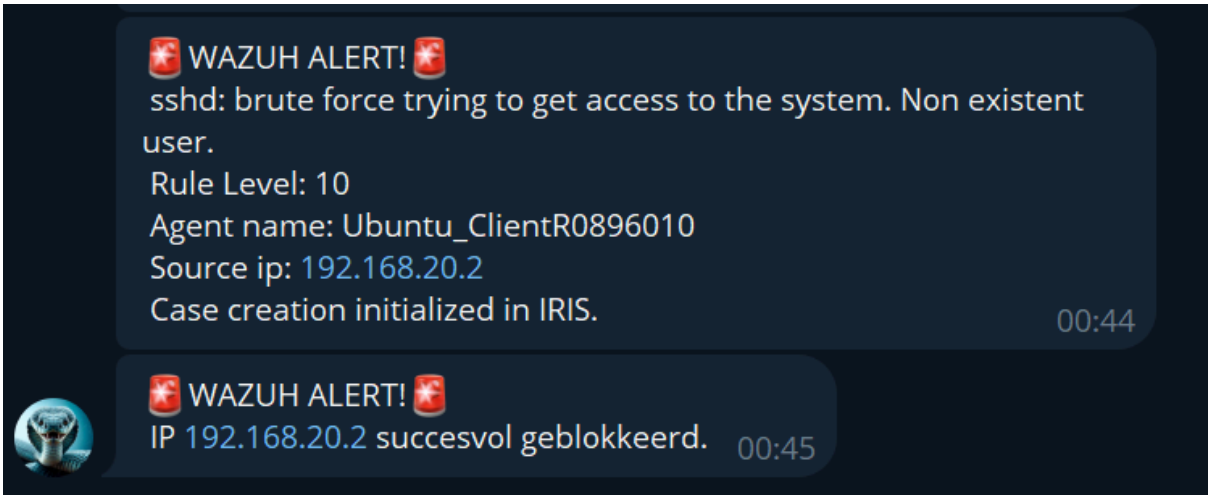
```python
# Simuleer de agentnaam uit de Wazuh-alert
agent_name = '$exec.all_fields.agent.name'  # Dit zou uit de Wazuh-alert komen

# Initialiseer case_customer
case_customer = 0  # Standaard klant-ID

# Controleer de agentnaam en stel case_customer in
if agent_name == "wazuhr0896010":
    case_customer = 2
elif agent_name == "Ubuntu_Clientr0896010":
    case_customer = 3
elif agent_name == "agentname3":
    case_customer = 4
else:
    case_customer = 0  # Standaard klant-ID als geen van de voorwaarden overeenkomt

# Print de gekozen case_customer voor controle
print(case_customer)
```

Bovenstaand script voor "customers" aan te wijzen voor IRIS.

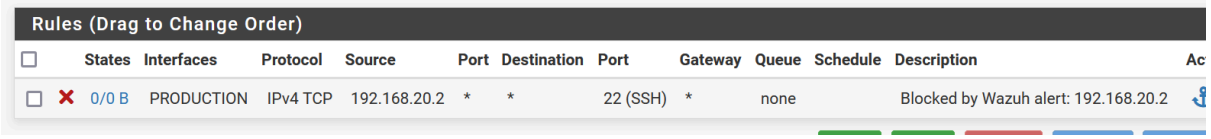
- **Automatic remediation via Pfsense API**: Als alerts betrekking hebben tot een bruteforce aanval dan wordt er in de telegram melding het **Source ip** weergegeven. Met een python script wordt dit **Source IP** opgevangen. En er gaat er een firewall rule aangemaakt worden in pfsense via de **API.** Alsook krijgen we een bevestigingsmelding in **Telegram.**

```python
import requests
import json

# API configuratie voor pfSense
PFSENSE_API_URL = "https://172.24.1.107:9999/api/v2/firewall/rule"
PFSENSE_API_KEY = "ccc33889040439711800643f2c866373"
rule_id_variabel = "$exec.rule_id"
source_ip_variabel = '$exec.all_fields.data.srcip'
rule_id_int = int(rule_id_variabel)
# Simuleer een Wazuh alert (dit zou in werkelijkheid uit een alert komen)
wazuh_alert = {
    "$exec.rule_id": rule_id_int,  # Bijvoorbeeld rule_id voor brute-force aanval
    "$exec.all_fields.data.srcip": source_ip_variabel,  # Bron-IP van de aanval
}

# Brute-force aanval rule_ids
BRUTE_FORCE_RULE_IDS = [5712, 5713, 5714]  # Voeg hier de relevante rule_ids toe

def block_ip(source_ip):
    """Voegt een firewallregel toe om het IP te blokkeren."""
    rule_payload = {
        "type": "block",  # Blokkeer actie
        "interface": ["SOC"],  # Specificeer de interface, bijvoorbeeld WAN
        "ipprotocol": "inet",  # Gebruik IPv4 (inet), je kunt 'inet6' gebruiken voor IPv6
        "protocol": "tcp",  # Specifieke blokkering voor TCP (SSH is een TCP-protocol)
        "source": source_ip,  # Het bron-IP dat we willen blokkeren
        "source_port": "22",  # Alle poorten van de bron
        "destination": "any",  # Alle bestemmingen
        "destination_port": "22",  # Blokkeer alleen poort 22 (SSH)
        "descr": f"Blocked by Wazuh alert: {source_ip}",  # Beschrijving van de regel
        "disabled": False,  # De regel is ingeschakeld
        "log": True,  # Log het blokkeringsactie
        "statetype": "keep state",  # Houd de staat van de verbinding bij
        "floating": True,  # Regel kan op elke interface toegepast worden
        "quick": True  # Regel is onmiddellijk van kracht, geen verdere regels worden gecontroleerd
    }

    try:
        # Verstuur de POST-aanvraag om de regel toe te voegen via de juiste endpoint
        response = requests.post(
            PFSENSE_API_URL,
            headers={
                "x-api-key": PFSENSE_API_KEY,
                "Content-Type": "application/json"
            },
            json=rule_payload,
            verify=False  # SSL-certificaatcontrole uitgeschakeld
        )

        if response.status_code == 200 or response.status_code == 201:
            return f"IP {source_ip} succesvol geblokkeerd."
        else:
            return f"Fout bij blokkeren van IP {source_ip}: {response.status_code}, {response.text}"

    except Exception as e:
        return f"Er is een fout opgetreden: {str(e)}"

def process_wazuh_alert(alert):
    """Verwerkt een Wazuh alert en blokkeert het IP als het een brute-force aanval betreft."""
    rule_id = alert["$exec.rule_id"]
    source_ip = alert["$exec.all_fields.data.srcip"]

    # Controleer of de rule_id behoort tot de brute-force aanvallen
    if rule_id in BRUTE_FORCE_RULE_IDS:
        return block_ip(source_ip)
    else:
        return f"Geen actie nodig voor rule_id {rule_id}."

# Test het verwerken van een alert
result = process_wazuh_alert(wazuh_alert)
print(result)
```

Script voor automatische FW regel aan te maken van het Source IP(aanvaller) in pfsense. (Bij Bruteforce aanval)



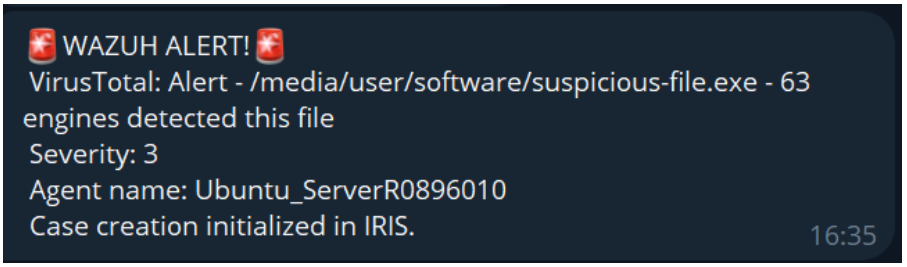
voorbeeld van melding tijdens een Bruteforce aanval. Plus bevestigingsmelding als het IP geblokkeerd is.



de pfsense FW regel die via de PFsense API is aangemaakt.

Threat Intelligence via FIM en VirusTotal:

- Wazuh maakt gebruik van **File Integrity Monitoring (FIM)** om wijzigingen in belangrijke systeem directories te detecteren.
- Bij verdachte wijzigingen wordt er via een integratie met **VirusTotal** gecontroleerd of de betrokken bestanden bekend zijn als schadelijk.

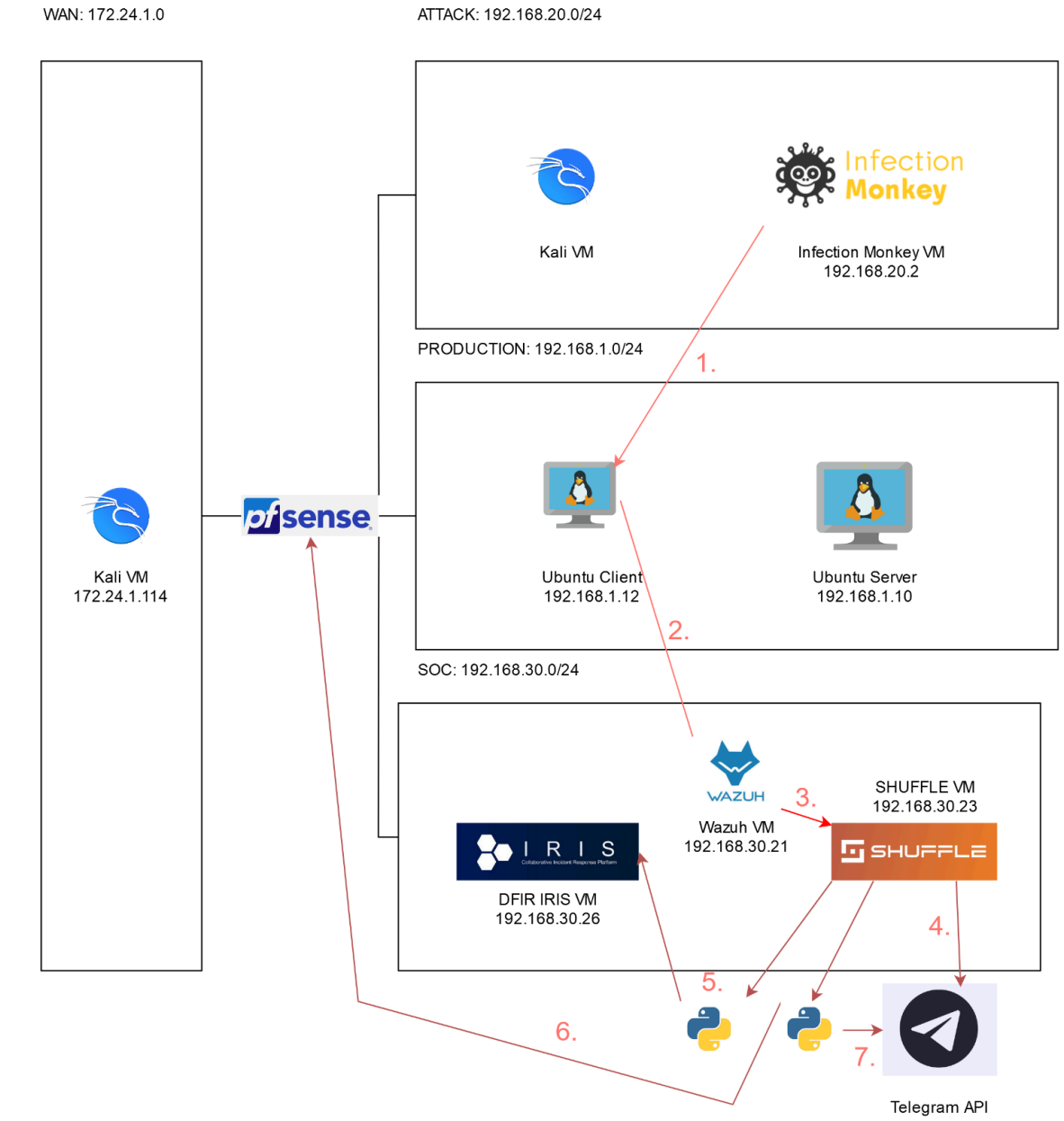


- Deze threat intelligence wordt gebruikt om snellere en nauwkeurigere beslissingen te nemen.

Infection Monkey
Voor de Audit tool gebruik ik Infection Monkey, deze gaan we gebruiken tijdens de Demo voor het demonstreren van een Bruteforce attack.

Diagram van Attack scenario 1

Aanval met **Infection Monkey**



1. Infection monkey begint een bruteforce aanval op Ubuntu client 192.168.1.12

2. Tentakels van Wazuh(OSSEC) merken verdachte activiteiten op, Alerts worden gegenereerd

3. Alerts worden doorgestuurd naar Shuffle. En de Shuffle workflow wordt getriggerd

4. Alert wordt gefiltered en verstuurd naar Telegram

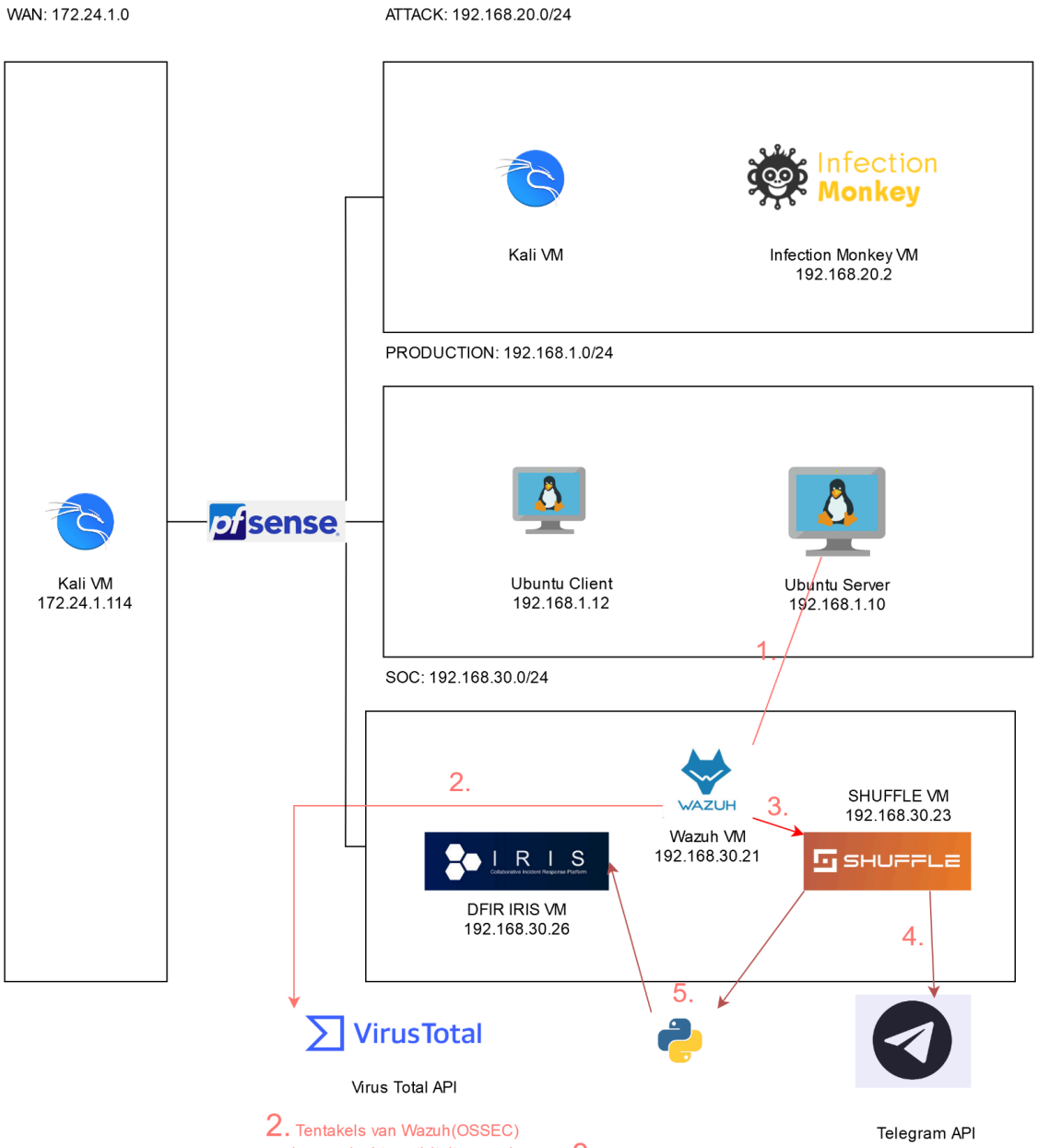
5. Melding wordt door python script gehaald om customer aan de case toe te wijzen, vervolgens wordt er een case aangemaakt in IRIS

6. Alert wordt door een python script gehaald, indien deze alert betrekking heeft tot Bruteforce, wordt het Source IP opgehaald en geblokkeerd in PFsense als een floating rule

7. Bevestiging van blokkeren IP wordt verstuurd naar Telegram.

Diagram Attack Scenario(Malware) 2

Download van malware door gebruiker.



1. Ubuntu Server download malware(virus)

2. Tentakels van Wazuh(OSSEC) merken verdachte activiteiten op via File Integrity Monitoring in een map dat wordt gemonitord door FIM. Dit bestand wordt gechecked via VirusTotal API

3. Alerts worden doorgestuurd naar Shuffle. En de Shuffle workflow wordt getriggerd

4. Alert wordt gefiltered en verstuurd naar Telegram

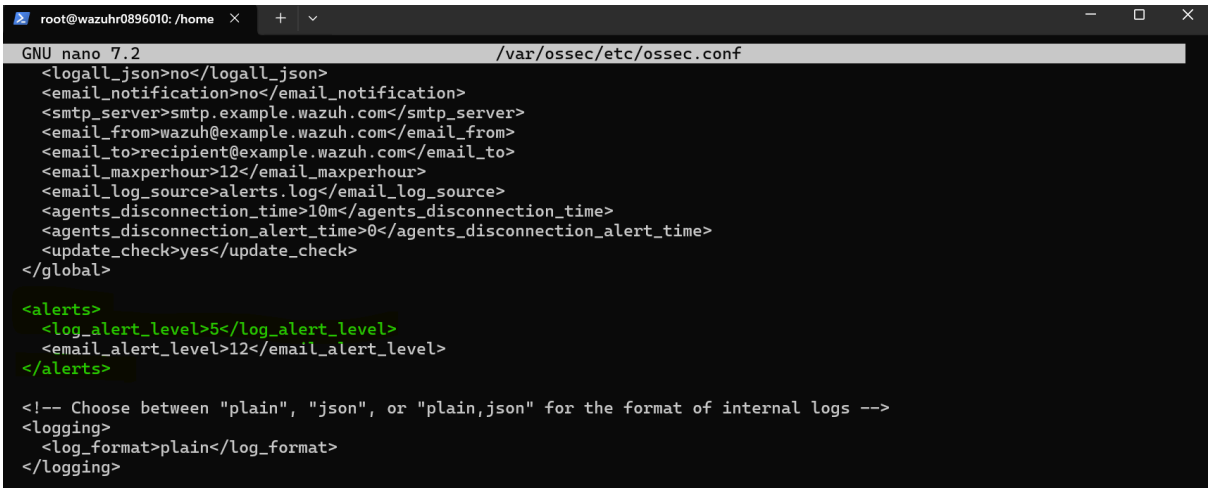
5. Melding wordt door python script gehaald om customer aan de case toe te wijzen, vervolgens wordt er een case aangemaakt in IRIS

Configureren Tools

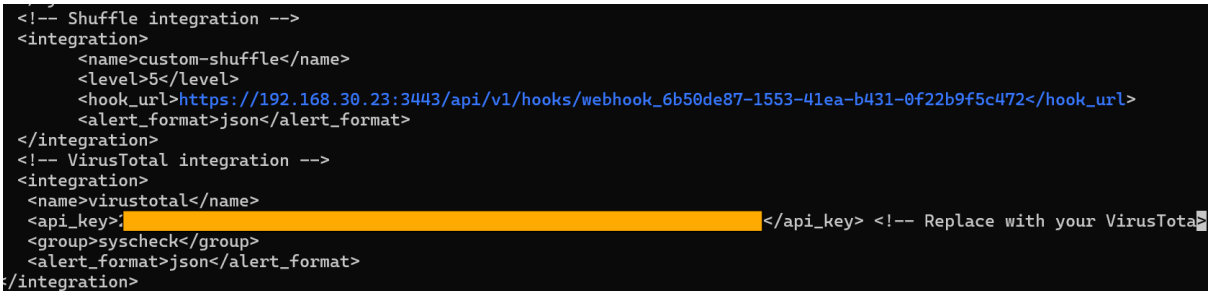
Wazuh

Voor de installatie heb ik de “stap voor stap installatie” gedaan om de tool al wat beter te leren kennen.
https://documentation.wazuh.com/current/installation-guide/index.html

Meeste configuraties binnen **Wazuh** bevinden zich in de **ossec.conf file**.



*Aanpassing van de “log alert level” naar **5.***



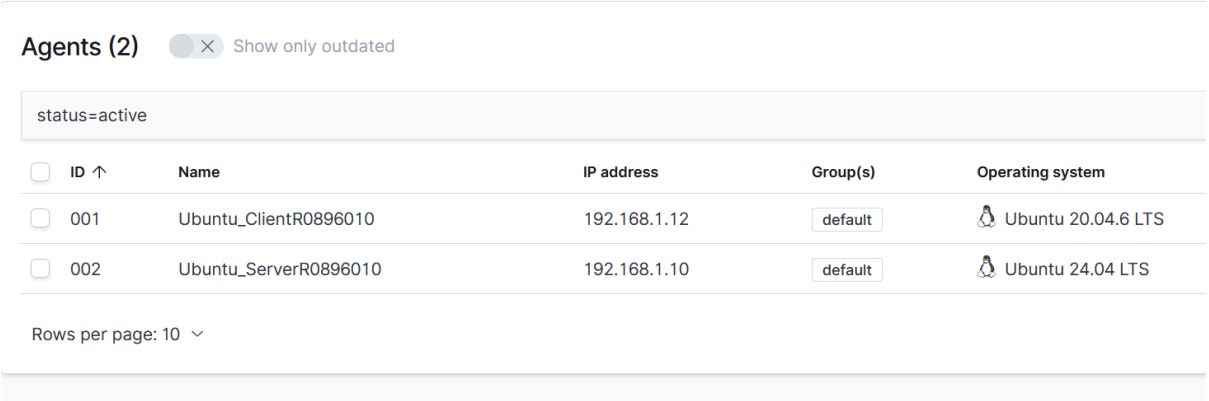
Integraties van “Shuffle” en van “Virustotal”

```
GNU nano 7.2                                    /var/ossec/etc/ossec.conf
<!-- 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>

  <!-- Directories to check  (perform all possible verifications) -->
  <directories>/etc,/usr/bin,/usr/sbin</directories>
  <directories>/bin,/sbin,/boot</directories>
  <directories check_all="yes" realtime="yes">/media/user/software</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>
```

*FIM monitoring **UbuntuServer** in **Production subnet.***

Toevoegen van Agents

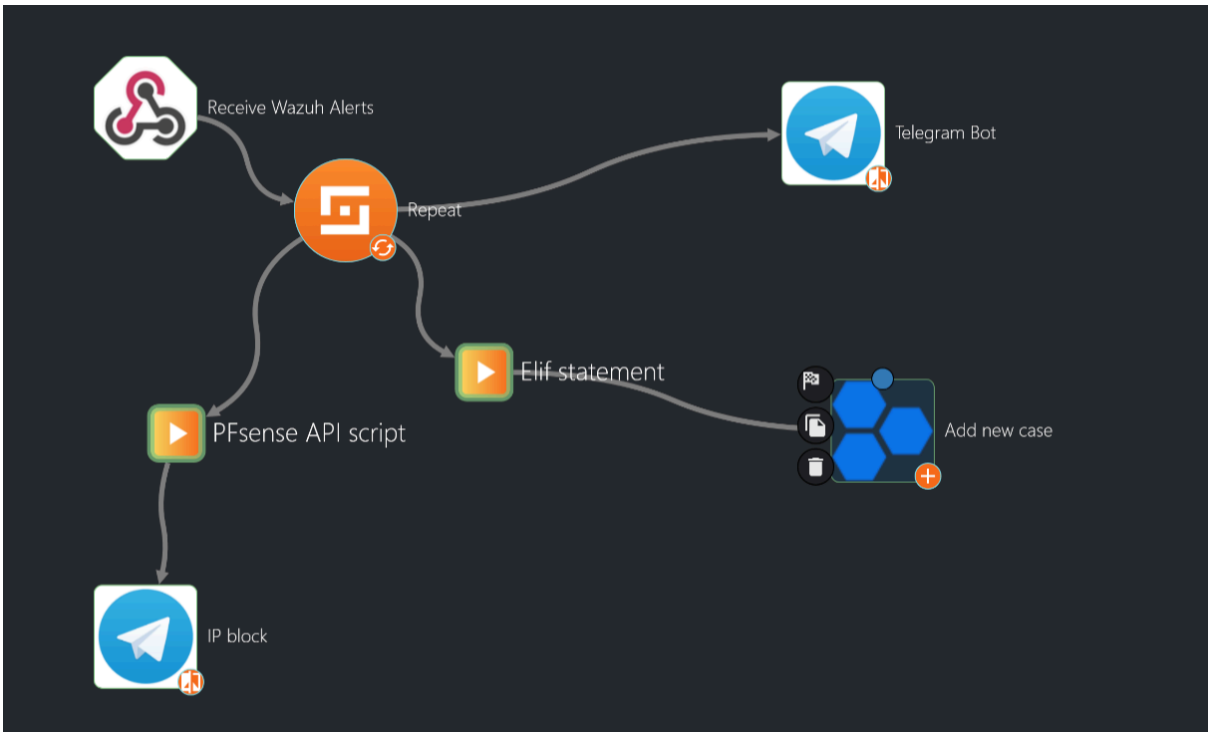
https://documentation.wazuh.com/current/installation-guide/wazuh-agent/wazuh-agent-package-linux.html



*Overzicht van de Agents die worden gemonitord. Deze bevinden zich in het **Production Subnet.***

Shuffle

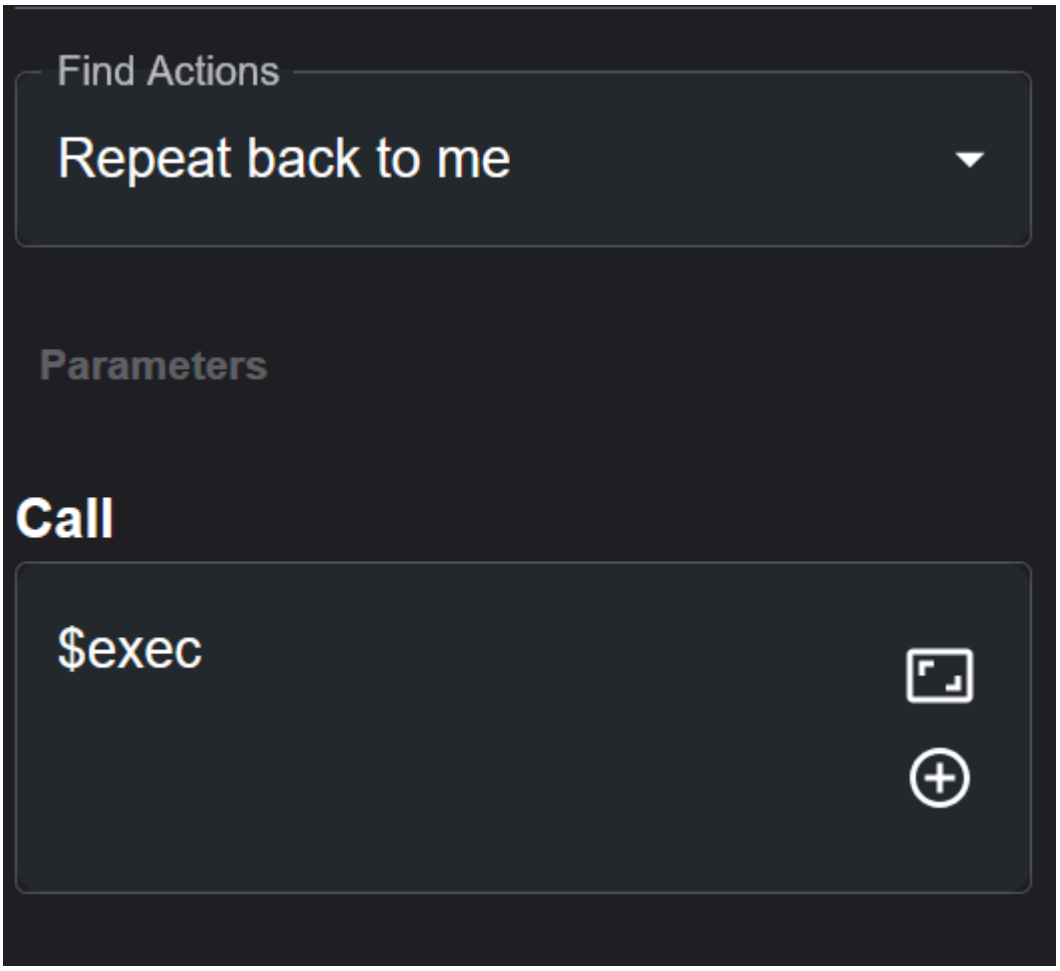
De workflow van Shuffle nader toegelicht.



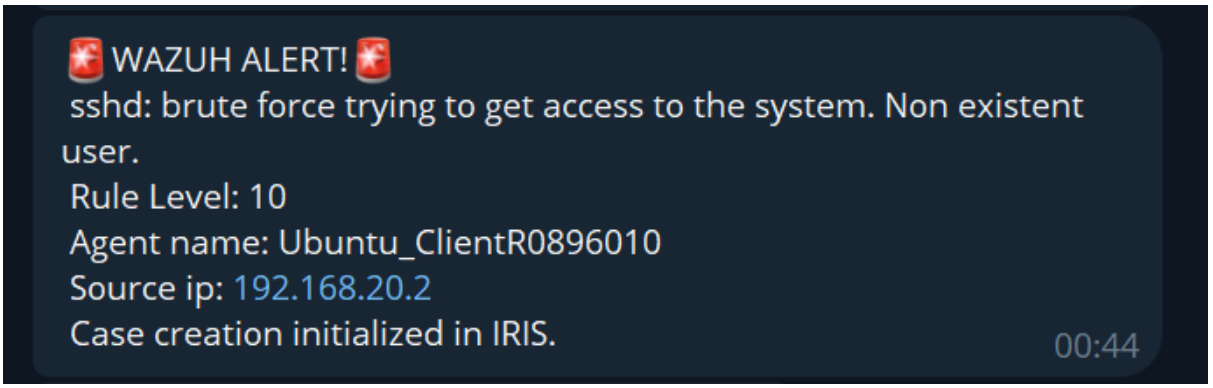
Workflow van "Shuffle".

De **Wazuh** alerts komen binnen via de webhook in shuffle.
(*https://172.24.1.107:3443/api/v1/hooks/webhook_6b50de87-1553-41ea-b431-0f22b9f5c472*)

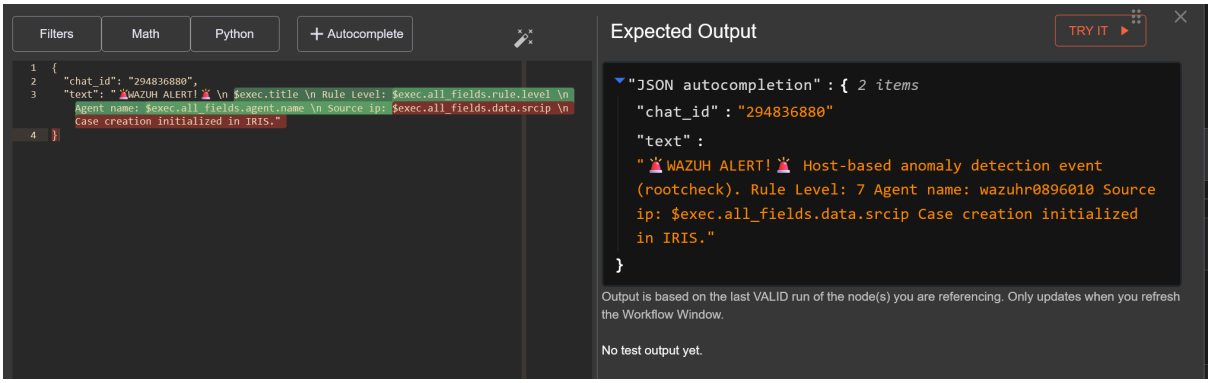
Deze wordt doorgestuurd naar een shuffle tool dat de alert herhaalt.



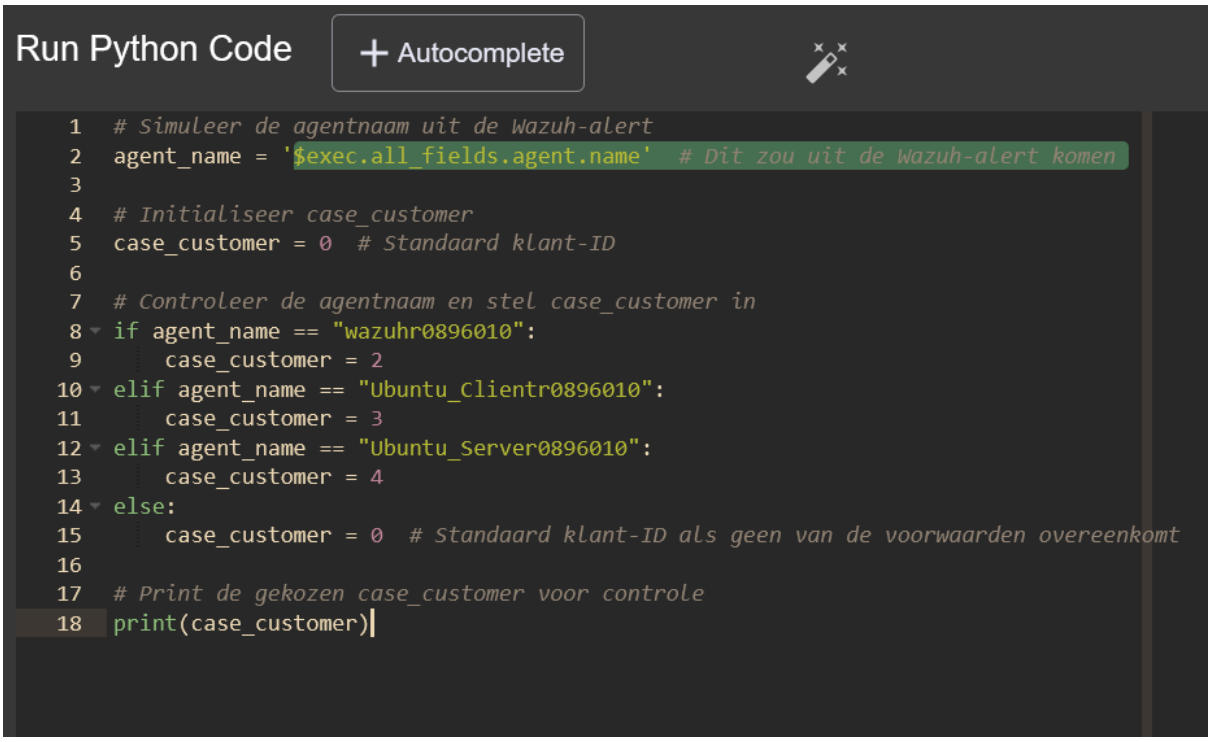
Deze Alert komt vervolgens binnen op *Telegram* en het “Customers Python script”



Telegram Alert.

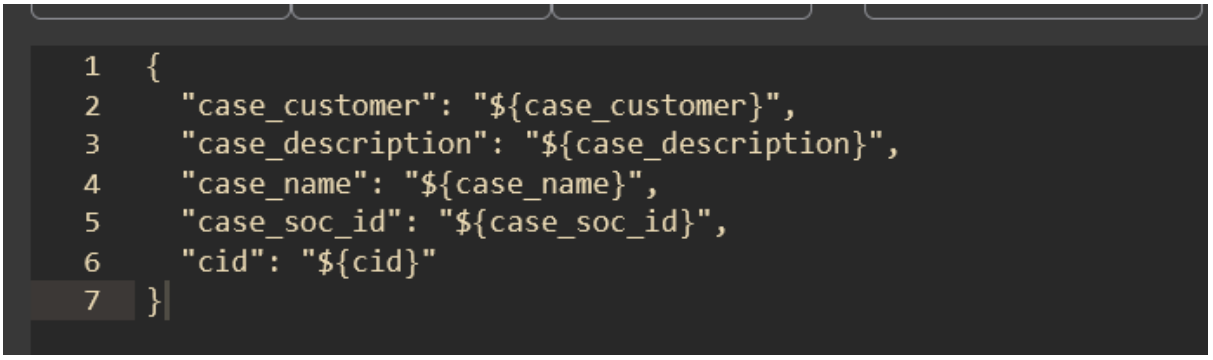


Telegram API



Python script voor het filteren van de “Customer” voor het aanmaken van het IRIS ticket.

Vervolgens wordt er automatisch in IRIS een ticket aangemaakt met de juiste “Customer” via een api van **IRIS**



API IRIS.

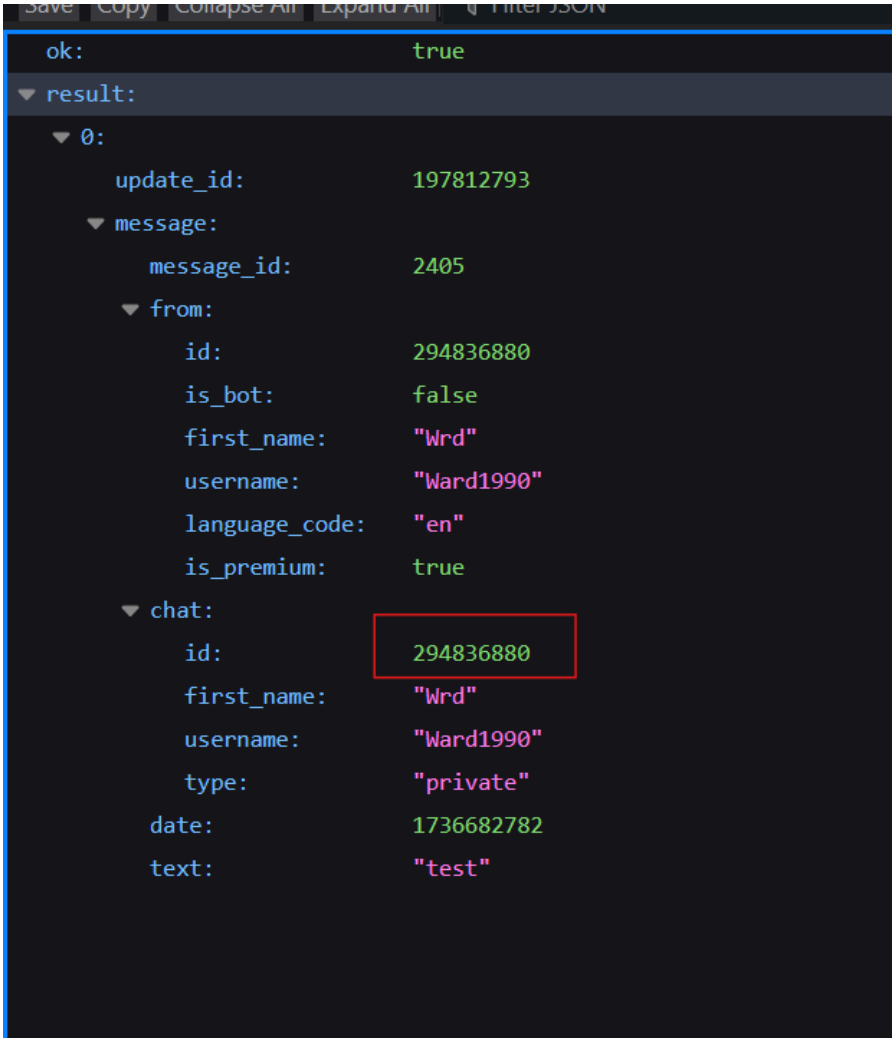
Telegram

Voor de alerts via Telegram heb ik gebruik gemaakt van **Botfather.** Dit is een Bot die je kan aanspreken via Telegram waarmee je dan zelf een bot kan maken. Tutorial grotendeels gevolgd via dit YT filmpje.

How to create a Telegram Bot with the Botfather

Eens de bot gemaakt is is het enkel via Shuffle nog even de URL meegeven en wat belangrijk is en niet zo heel duidelijk is gedocumenteerd binnen shuffle is dat je jou chat_id moet meegeven van jou telegram Bot.

Kan je vinden door volgende url in te geven in jou browser.
https://api.telegram.org/bot<YOUR_BOT_TOKEN>/getUpdates



Chat ID Telegram.

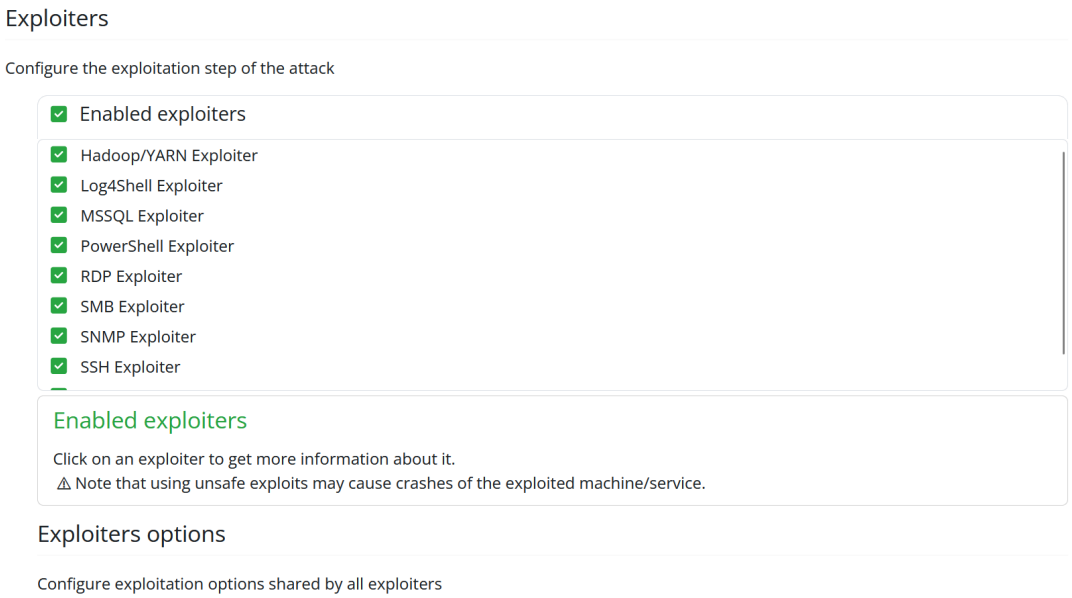
Iris

Installeren van Iris is gebeurd met docker via volgende link:
https://docs.dfir-iris.org/getting_started/

Infection Monkey

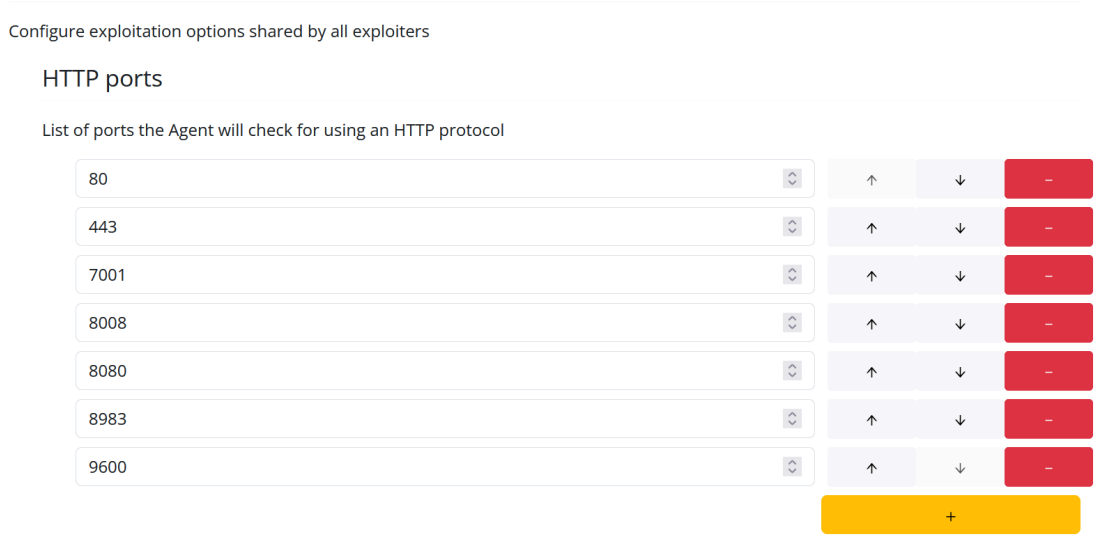
Installatie van infection monkey gebeurde via volgende link:
https://techdocs.akamai.com/infection-monkey/docs/linux

Voor de configuratie in **infection monkey** heb ik volgende aanpassingen gemaakt in de **configuration tab.**

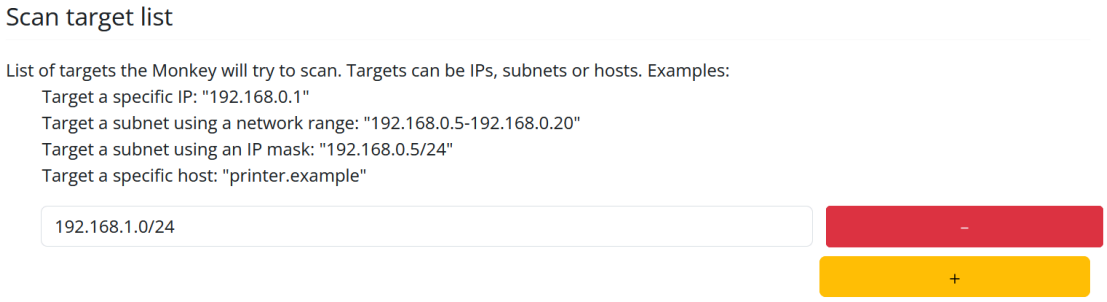


Exploiters aangeduid

Exploiters geinstalleerd en opgezet. En poorten (stond default) geselecteerd.

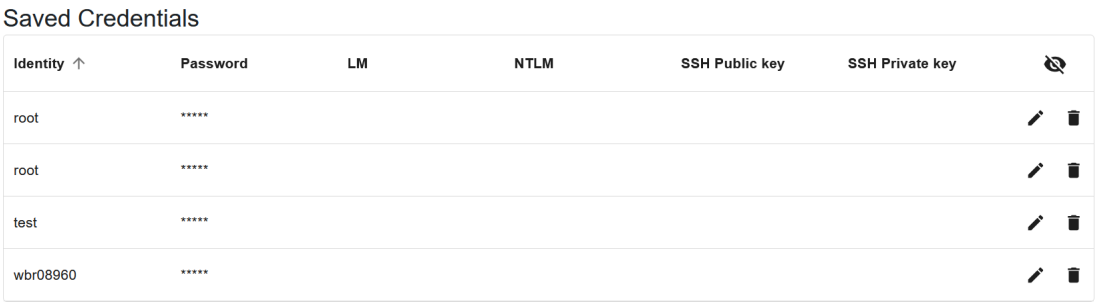


poorten die mogen gebruikt worden



scan target lists.

Het netwerk(production) dat mag gescanned en aangevallen worden.



Credential list

Lijst met credentials die mogen gebruikt worden voor de bruteforce.

Conclusie

De opzet van dit SOC toont een robuuste en geïntegreerde benadering van netwerkbeveiliging en incidentmanagement. Door gebruik te maken van tools zoals Wazuh voor SIEM, Ossec voor HIDS, en Shuffle voor automatisering, wordt een efficiënte en reactieve beveiligingsinfrastructuur gecreëerd. De configuratie van pfSense als firewall/router, samen met de specifieke subnetwerken, zorgt voor een goed afgeschermde en gemonitorde omgeving. Automatisering van alertverwerking en incidentrespons door middel van scripts en API's bewijst de effectiviteit van deze setup in het snel detecteren en mitigeren van bedreigingen. Dit systeem biedt een sterke basis voor verdere uitbreiding en verfijning van beveiligingspraktijken in een dynamische IT-omgeving