# SAE5.Cyber.03 – TP Wazuh

Dans un contexte où la sécurité des systèmes d'information devient un enjeu majeur, la capacité à superviser les équipements réseau et serveurs est essentielle pour détecter rapidement les incidents et prévenir les attaques. Les infrastructures informatiques actuelles génèrent une quantité importante de logs qu'il est nécessaire de centraliser, d'analyser et de corréler afin d'obtenir une vision claire de l'activité du réseau.

Ce projet a pour objectif de concevoir et mettre en place une solution complète de supervision et de gestion des logs, permettant de suivre l'état des différents équipements du réseau, de repérer les comportements anormaux et de réagir efficacement en cas d'incident.

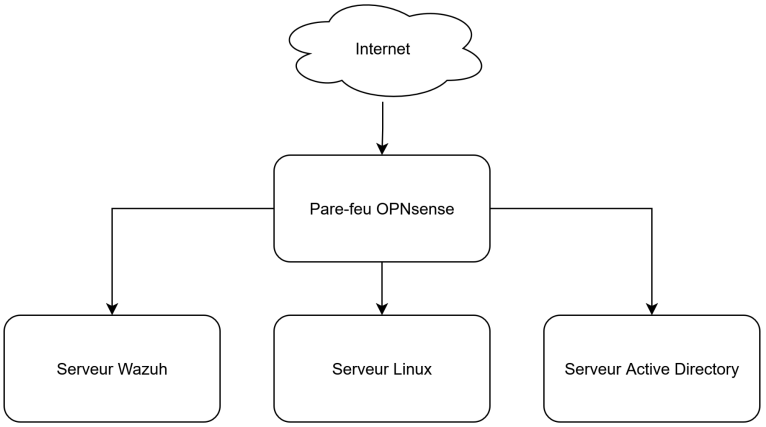
Ce rapport présente les différentes étapes de la mise en œuvre de cette solution : de la définition de l'infrastructure et du plan d'adressage à la configuration des outils de supervision, en passant par l'intégration d'un système d'alerte et la vérification de son bon fonctionnement.

## *SOMMAIRE*

# 1 - Infrastructure et table d'adressage

L'infrastructure mise en place repose sur un réseau local composé de plusieurs serveurs interconnectés. L'ensemble est organisé autour d'un pare-feu OPNsense, qui joue à la fois le rôle de passerelle vers Internet et de filtre de sécurité grâce à ses fonctions de WAF (Web Application Firewall) et d'IDS/IPS (détection et prévention d'intrusion).

Le réseau comprend les éléments suivants :




- Un serveur Active Directory, pour la gestion centralisée des utilisateurs et des politiques de sécurité.
- Un serveur Linux (Debian ou Ubuntu), hébergeant les services web (Apache) et base de données (MySQL).
- Un serveur Wazuh, chargé de la centralisation et de l'analyse des journaux d'activité.
- Le pare-feu OPNsense, garantissant la sécurité et la remontée des logs vers Wazuh.

Les différents éléments sont adressés de la manière suivante :

| Nom du serveur | Adresse IP | Passerelle par défaut |
| --- | --- | --- |
| Active Directory | 192.168.10.20/24 | 192.168.10.1 |
| OPNsense Firewall | 192.168.10.1/24 | x |
| Serveur Linux (Debian) | 192.168.10.15/24 | 192.168.10.1 |
| Serveur Wazuh | 192.168.10.11/24 | 192.168.10.1 |

Le pare-feu OPNsense assure la liaison entre le réseau interne et Internet, permettant l'installation des composants nécessaires tout en filtrant le trafic pour garantir la sécurité globale de l'infrastructure.

## 2 - Installation et configuration du serveur Wazuh

Nous commençons par importer la clé GPG de Wazuh afin de pouvoir authentifier les paquets téléchargés depuis le dépôt officiel :

```
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | gpg
--no-default-keyring --keyring gnupg-ring:/usr/share/keyrings/wazuh.gpg
--import && chmod 644 /usr/share/keyrings/wazuh.gpg
```

```
root@debian:~# curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | gpg --no-d
efault-keyring --keyring gnupg-ring:/usr/share/keyrings/wazuh.gpg --import && ch
mod 644 /usr/share/keyrings/wazuh.gpg
gpg: le porte-clefs « /usr/share/keyrings/wazuh.gpg » a été créé
gpg: répertoire « /root/.gnupg » créé
gpg: /root/.gnupg/trustdb.gpg : base de confiance créée
gpg: clef 96B3EE5F29111145 : clef publique « Wazuh.com (Wazuh Signing Key) <supp
ort@wazuh.com> » importée
gpg:       Quantité totale traitée : 1
gpg:                   importées : 1
```

Nous ajoutons ensuite le dépôt Wazuh à la liste des sources APT du système :

```
echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg]
https://packages.wazuh.com/4.x/apt/ stable main" | tee -a
/etc/apt/sources.list.d/wazuh.list
```

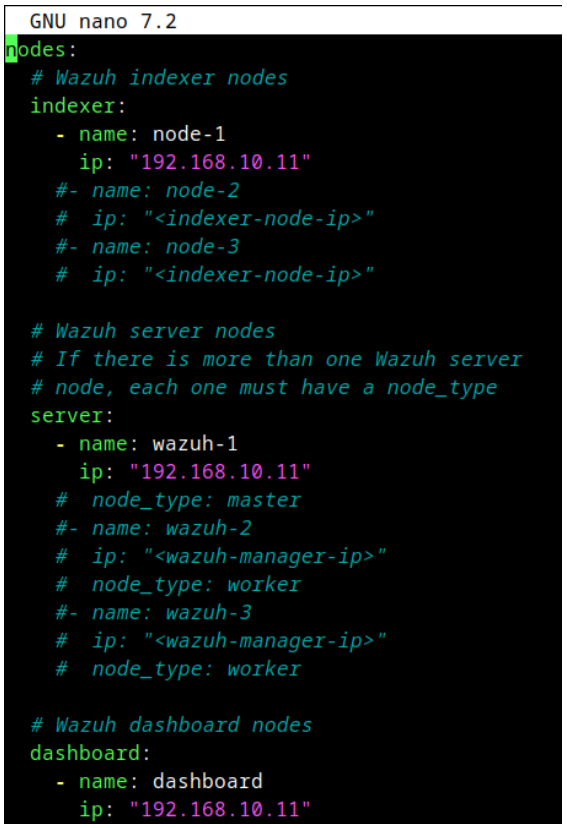
Puis nous rafraîchissons les dépôts et installons les dépendances nécessaires :

```
sudo apt update && sudo apt upgrade && sudo apt-get install debconf
adduser procps curl gnupg apt-transport-https filebeat debhelper
libcap2-bin
```

Nous téléchargeons le script de génération des certificats et son fichier de configuration :

```
curl -sO https://packages.wazuh.com/4.7/wazuh-certs-tool.sh && curl -sO
https://packages.wazuh.com/4.7/config.yml
```

Nous modifions le fichier config.yml pour l'adapter à nos adresses IP et les noms des nœuds :



```
 GNU nano 7.2
nodes:
  # Wazuh indexer nodes
  indexer:
    - name: node-1
      ip: "192.168.10.11"
    #- 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: "192.168.10.11"
    #  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: "192.168.10.11"
```

Nous lançons ensuite la génération des certificats nécessaires à la communication sécurisée entre les composants Wazuh :

```
bash ./wazuh-certs-tool.sh -A
tar -cvf ./wazuh-certificates.tar -C ./wazuh-certificates/ .
rm -rf ./wazuh-certificates
```

```
root@debian:~# bash ./wazuh-certs-tool.sh -A
17/09/2025 08:49:42 INFO: Admin certificates created.
17/09/2025 08:49:42 INFO: Wazuh indexer certificates created.
17/09/2025 08:49:42 INFO: Wazuh server certificates created.
17/09/2025 08:49:42 INFO: Wazuh dashboard certificates created.
```

Puis nous installons les paquets principaux de Wazuh :

```
sudo apt install wazuh-indexer wazuh-manager wazuh-dashboard -y
```

Nous déployons dans le répertoire de l'indexer les certificats générés précédemment :

```
NODE_NAME=node-1
mkdir /etc/wazuh-indexer/certs
tar -xf ./wazuh-certificates.tar -C /etc/wazuh-indexer/certs/
./$NODE_NAME.pem ./$NODE_NAME-key.pem ./admin.pem ./admin-key.pem
./root-ca.pem
mv -n /etc/wazuh-indexer/certs/$NODE_NAME.pem
/etc/wazuh-indexer/certs/indexer.pem
mv -n /etc/wazuh-indexer/certs/$NODE_NAME-key.pem
/etc/wazuh-indexer/certs/indexer-key.pem
chmod 500 /etc/wazuh-indexer/certs
chmod 400 /etc/wazuh-indexer/certs/*
chown -R wazuh-indexer:wazuh-indexer /etc/wazuh-indexer/certs
```

Puis nous activons et démarrons le service Wazuh Indexer :

```
systemctl daemon-reload
systemctl enable wazuh-indexer
systemctl start wazuh-indexer
```

Nous vérifions le bon fonctionnement du cluster :

```
bash /usr/share/wazuh-indexer/bin/indexer-security-init.sh
curl -k -u admin:admin https://192.168.1.73:9200
curl -k -u admin:admin https://192.168.1.73:9200/_cat/nodes?v
```

Puis nous activons, démarrons et vérifions le service Wazuh Indexer :

```
systemctl daemon-reload
systemctl enable wazuh-manager
systemctl start wazuh-manager
systemctl status wazuh-manager
```

Nous passons ensuite à l'installation de Filebeat, utilisé pour le transfert des logs :

```
apt install filebeat
curl -so /etc/filebeat/filebeat.yml
https://packages.wazuh.com/4.7/tpl/wazuh/filebeat/filebeat.yml
nano /etc/filebeat/filebeat.yml
filebeat keystore create
```

Nous ajoutons les identifiants par défaut au keystore :

```
echo admin | filebeat keystore add username --stdin --force
echo admin | filebeat keystore add password --stdin --force
```

Nous téléchargeons un modèle d'alerte et les modules supplémentaires Filebeat :

```
curl -so /etc/filebeat/wazuh-template.json
https://raw.githubusercontent.com/wazuh/wazuh/v4.7.2/extensions/elastics
earch/7.x/wazuh-template.json
chmod go+r /etc/filebeat/wazuh-template.json
curl -s
https://packages.wazuh.com/4.x/filebeat/wazuh-filebeat-0.3.tar.gz | tar
-xvz -C /usr/share/filebeat/module
```

Puis nous déployons les certificats Filebeat :

```
NODE_NAME=node-1
mkdir /etc/filebeat/certs
tar -xf ./wazuh-certificates.tar -C /etc/filebeat/certs/
./$NODE_NAME.pem ./$NODE_NAME-key.pem ./root-ca.pem
mv -n /etc/filebeat/certs/$NODE_NAME.pem
/etc/filebeat/certs/filebeat.pem
mv -n /etc/filebeat/certs/$NODE_NAME-key.pem
/etc/filebeat/certs/filebeat-key.pem
chmod 500 /etc/filebeat/certs
chmod 400 /etc/filebeat/certs/*
chown -R root:root /etc/filebeat/certs
```

Nous activons et démarrons le service Filebeat :

```
systemctl daemon-reload
systemctl enable filebeat
systemctl start filebeat
filebeat test output
```

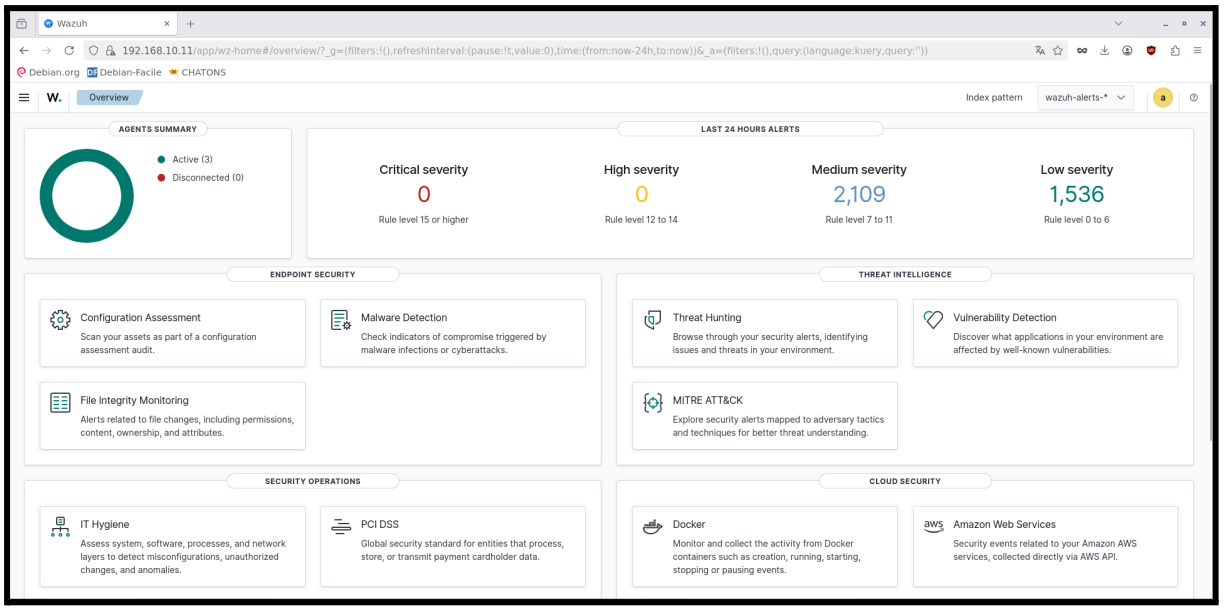
Puis nous déployons les certificats et activons le service du dashboard :

```
NODE_NAME=node-1
mkdir /etc/wazuh-dashboard/certs
tar -xf ./wazuh-certificates.tar -C /etc/wazuh-dashboard/certs/
./$NODE_NAME.pem ./$NODE_NAME-key.pem ./root-ca.pem
mv -n /etc/wazuh-dashboard/certs/$NODE_NAME.pem
/etc/wazuh-dashboard/certs/dashboard.pem
mv -n /etc/wazuh-dashboard/certs/$NODE_NAME-key.pem
/etc/wazuh-dashboard/certs/dashboard-key.pem
chmod 500 /etc/wazuh-dashboard/certs
chmod 400 /etc/wazuh-dashboard/certs/*
chown -R wazuh-dashboard:wazuh-dashboard /etc/wazuh-dashboard/certs
systemctl daemon-reload
systemctl enable wazuh-dashboard
systemctl start wazuh-dashboard
```

Nous modifions le fichier ossec.conf afin d'autoriser l'affichage de tous les types de logs :

```
  GNU nano 7.2                                        /var/ossec/etc/ossec.conf
<!--
  Wazuh - Manager - Default configuration for debian 12
  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>yes</logall>
    <logall_json>yes</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>15m</agents_disconnection_time>
    <agents_disconnection_alert_time>0</agents_disconnection_alert_time>
    <update_check>yes</update_check>
  </global>
```

Et une fois ces étapes terminées, nous pouvons accéder à l'interface web Wazuh :




L'infrastructure Wazuh est désormais opérationnelle et accessible via l'interface web, prête à recevoir les logs des différents agents du réseau pour assurer la supervision centralisée de notre système.

## 3 - Installation de l'agent Wazuh sur les autres équipements

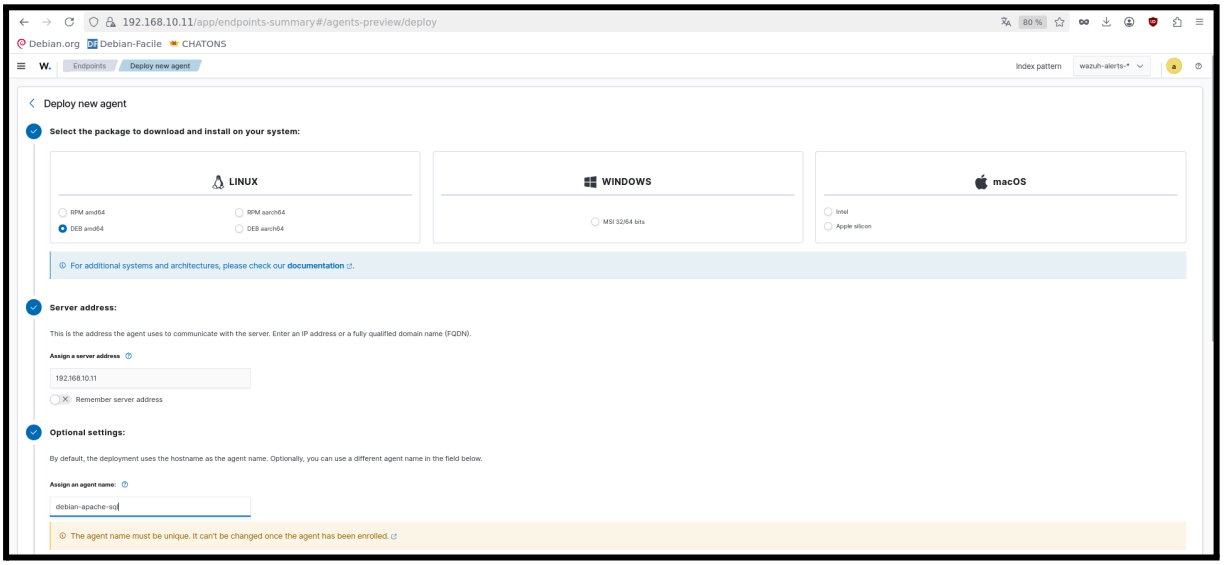
Pour compléter la supervision centralisée, nous installons les agents Wazuh sur les différents serveurs du réseau. Ces agents permettent de collecter et transmettre les logs locaux vers le serveur Wazuh, afin de centraliser les événements et faciliter l'analyse.

Nous commençons par autoriser les communications distantes avec Wazuh (donc sur le port par défaut 1514) en utilisant TCP :

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

Ensuite, nous ajoutons les agents Wazuh sur nos autres serveurs.

Pour le serveur Debian, nous utilisons l'interface web de Wazuh, qui propose une option permettant de générer la commande d'installation adaptée à chaque système d'exploitation. Nous commençons donc par renseigner les informations nécessaires :




Concrètement, nous sélectionnons le système d'exploitation du serveur, renseignons l'adresse IP du serveur Wazuh et attribuons un nom à l'agent avant de générer la commande d'installation.

Et la commande est donc correctement générée et présentée dans la partie 4 :


Voici la commande de la partie 4 :

```
wget
https://packages.wazuh.com/4.x/apt/pool/main/w/wazuh-agent/wazuh-agent_4
.13.1-1_amd64.deb && sudo WAZUH_MANAGER='192.168.10.11'
WAZUH_AGENT_NAME='debian-apache-sql' dpkg -i
./wazuh-agent_4.13.1-1_amd64.deb
```

Dans la partie 5, il y a également trois commandes fournies qui permettent de lancer l'agent :

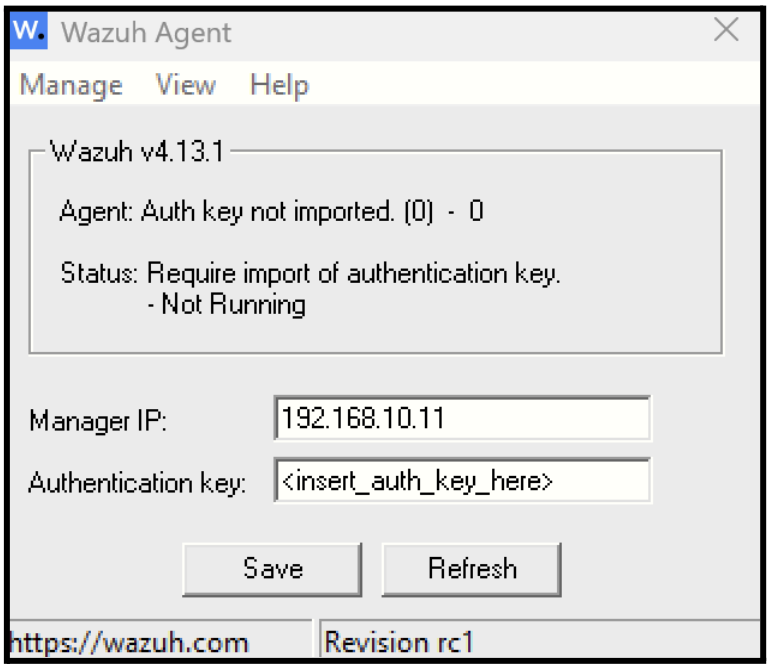
```
sudo systemctl daemon-reload
sudo systemctl enable wazuh-agent
sudo systemctl start wazuh-agent
```

Nous exécutons donc toutes ces commandes.

Puis nous installons l'agent pour le serveur Active Directory, la procédure est similaire, mais entièrement graphique, nous commençons par télécharger l'installateur de l'agent wazuh sur Windows via l'URL suivante :

https://packages.wazuh.com/4.x/windows/wazuh-agent-4.13.1-1.msi

L'installateur Wazuh pour Windows gère automatiquement la configuration après que nous lui ayons renseigné l'adresse IP du serveur Wazuh et la clé d'authentification :




Enfin, nous installons l'agent Wazuh sur l'OPNsense au moyen de la commande suivante :

```
pkg install os-wazuh-agent
```

Puis configurons le fichier ossec.conf afin que l'agent puisse communiquer avec le serveur :

```
root@OPNsense:~ # cat /var/ossec/etc/ossec.conf
<ossec_config>
  <client>
    <server>
      <address>192.168.10.11</address>
      <protocol>tcp</protocol>
      <port>1514</port>
    </server>
    <crypto_method>aes</crypto_method>
    <enrollment>
      <port>1515</port>
    </enrollment>
  </client>
```

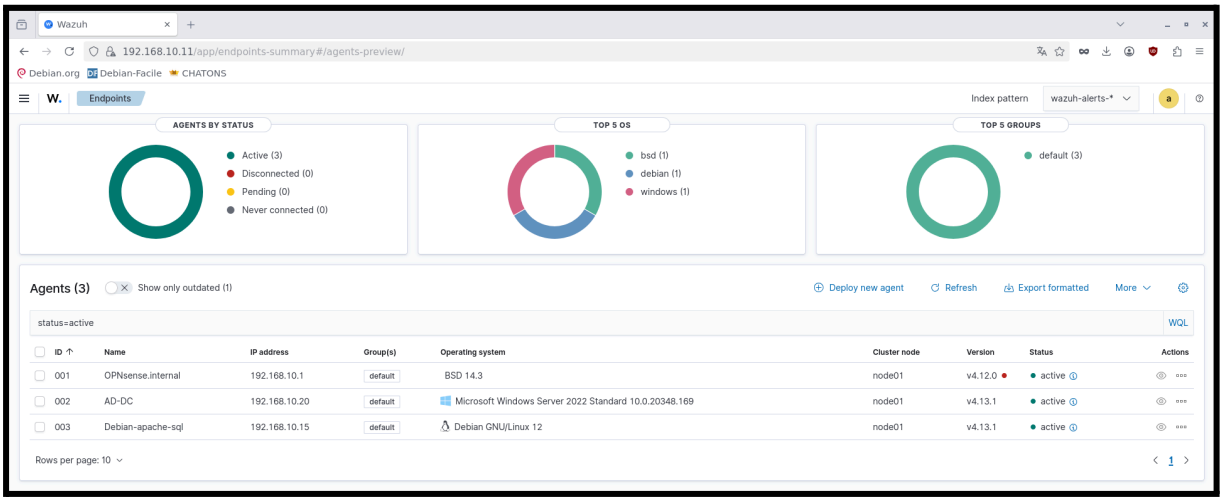
Il faut également que l'agent Wazuh sache quels sont les fichiers de logs pertinents à envoyer au serveur :

```
<!-- Suricata -->
<localfile>
  <log_format>json</log_format>
  <location>/var/log/suricata/eve.json</location>
</localfile>


<!-- Log analysis -->
<localfile>
  <log_format>syslog</log_format>
  <location>/var/ossec/logs/active-responses.log</location>
</localfile>

<localfile>
  <log_format>syslog</log_format>
  <location>/var/ossec/logs/opnsense_syslog.log</location>
</localfile>
```

Puis, avant de pouvoir observer les résultats sur le serveur Wazuh, il faut d'abord redémarrer le service wazuh-manager au moyen de la commande suivante :

```
sudo systemctl restart wazuh-manager
```

Et nous pouvons à présent voir les informations récoltées par l'agent Wazuh de l'OPNsense, ainsi que des serveurs Linux et Active Directory sur l'interface web du serveur Wazuh :



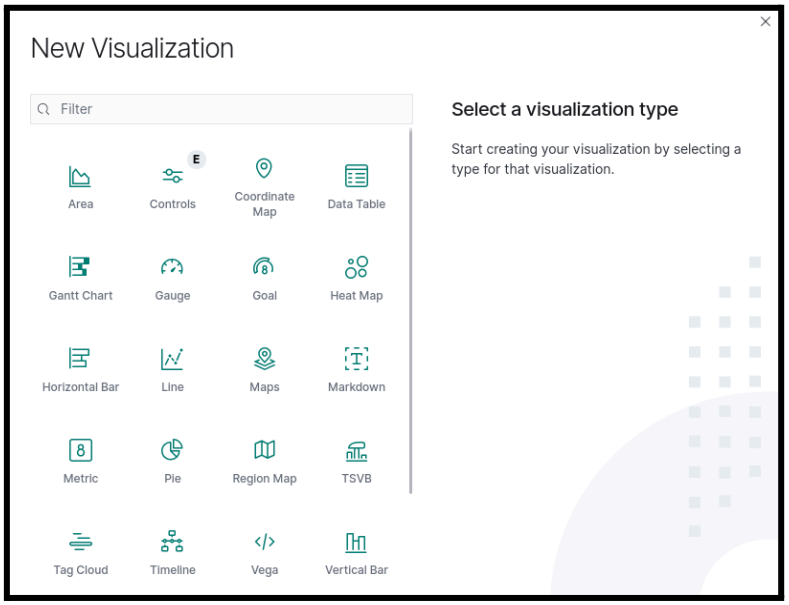

Nous pouvons bien voir que le statut actif des trois agents confirme leur fonctionnement.

## 4 - Création du dashboard Wazuh

Une fois les agents installés et les logs correctement remontés vers le serveur Wazuh, nous mettons en place un dashboard personnalisé afin de visualiser et d'analyser en temps réel les journaux collectés.

Pour cela, nous utilisons l'outil intégré de Wazuh permettant de créer des tableaux de bord à partir des logs. Chaque panneau affichera les événements d'un équipement spécifique, avec un filtrage adapté et une actualisation automatique pour un suivi continu de l'activité du réseau.

Nous commençons par créer une visualisation de type Data Table (Tableau de Données) :

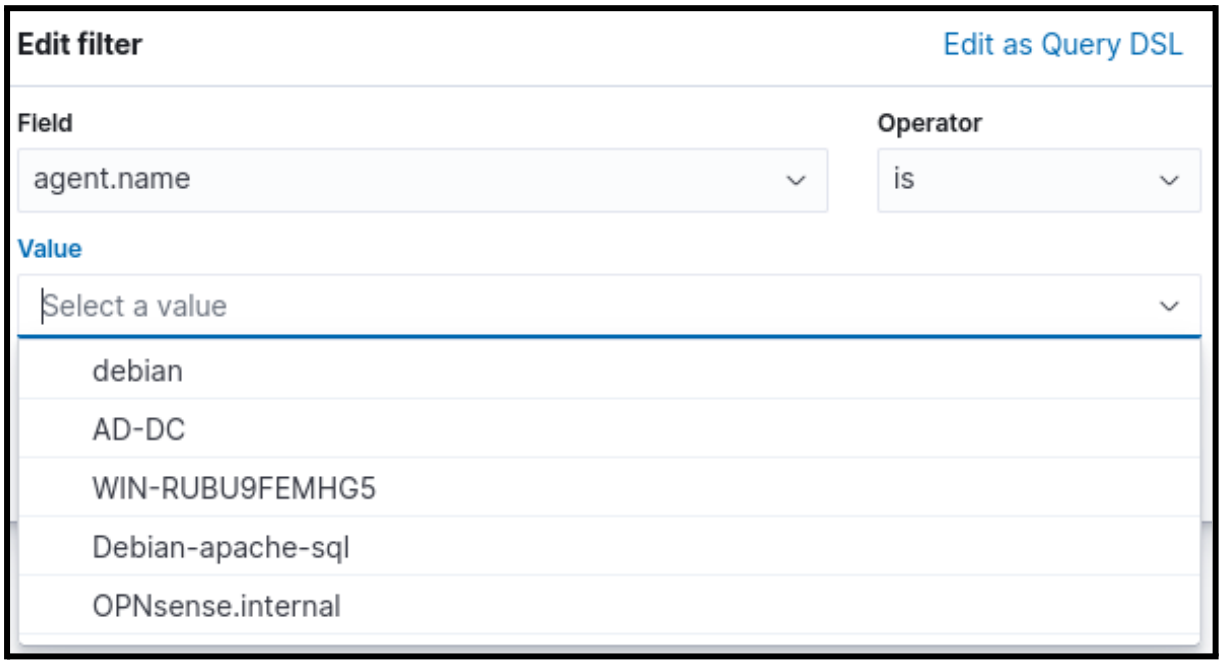



Nous choisissons le type des données qui seront visualisées, dans notre cas wazuh-alerts-* puisque nous voulons observer des logs :

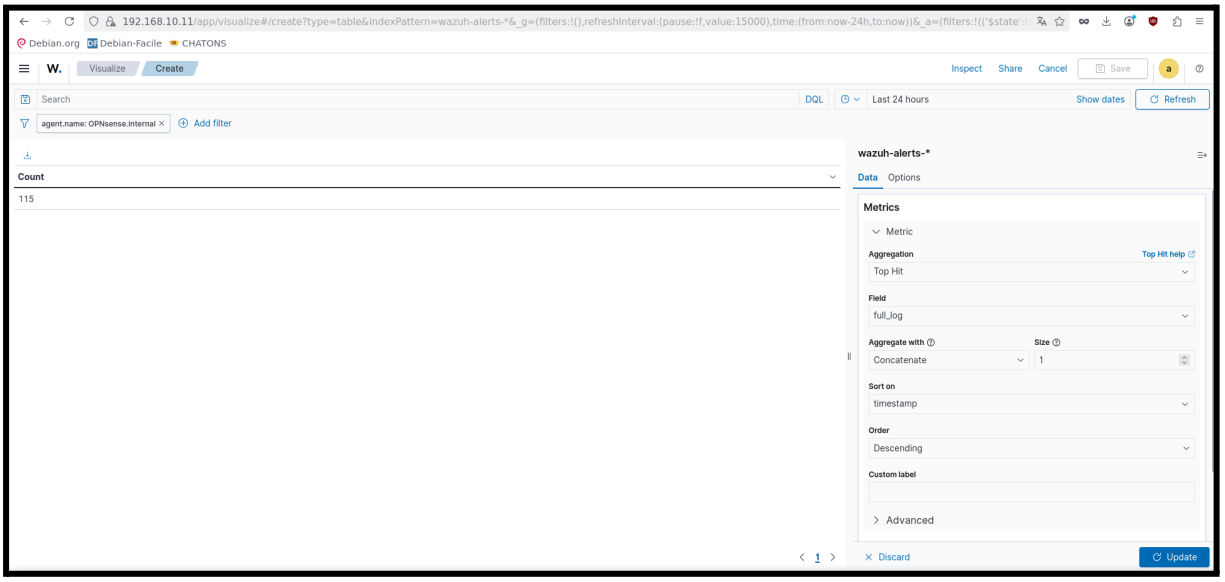
Après avoir sélectionné l'index wazuh-alerts-*, la visualisation affiche une valeur de count correspondant au nombre total de logs enregistrés par Wazuh. Dans notre cas, les 5472 counts indiquent que le serveur a collecté 5472 événements ou alertes provenant des différents agents supervisés :


Le dashboard s'affiche, mais aucun log n'est encore visible : seule la métrique count apparaît, additionnant tous les logs des différentes machines.

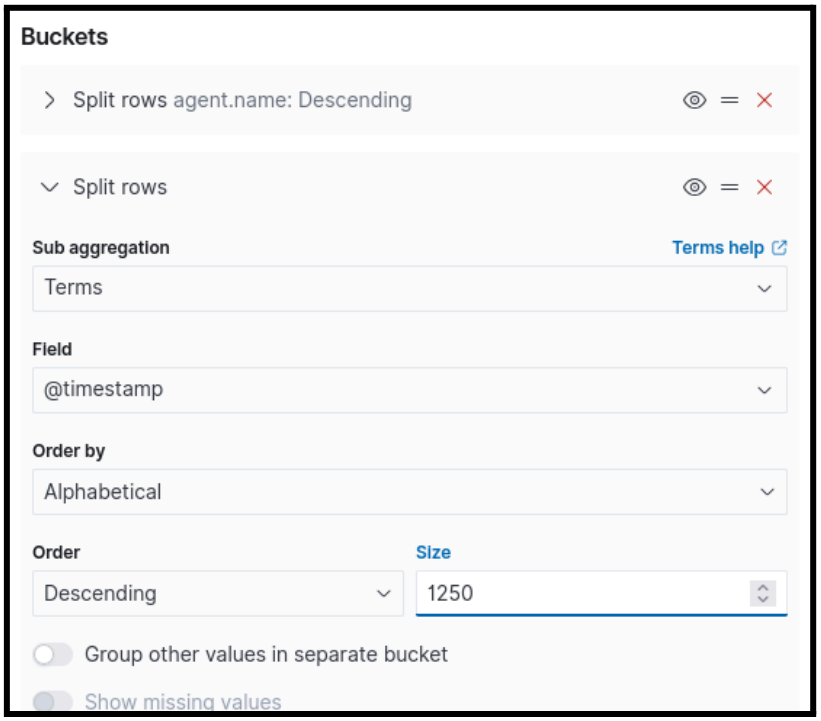
Nous ajoutons un filtre sur agent.name et choisissons l'agent concerné dans value :



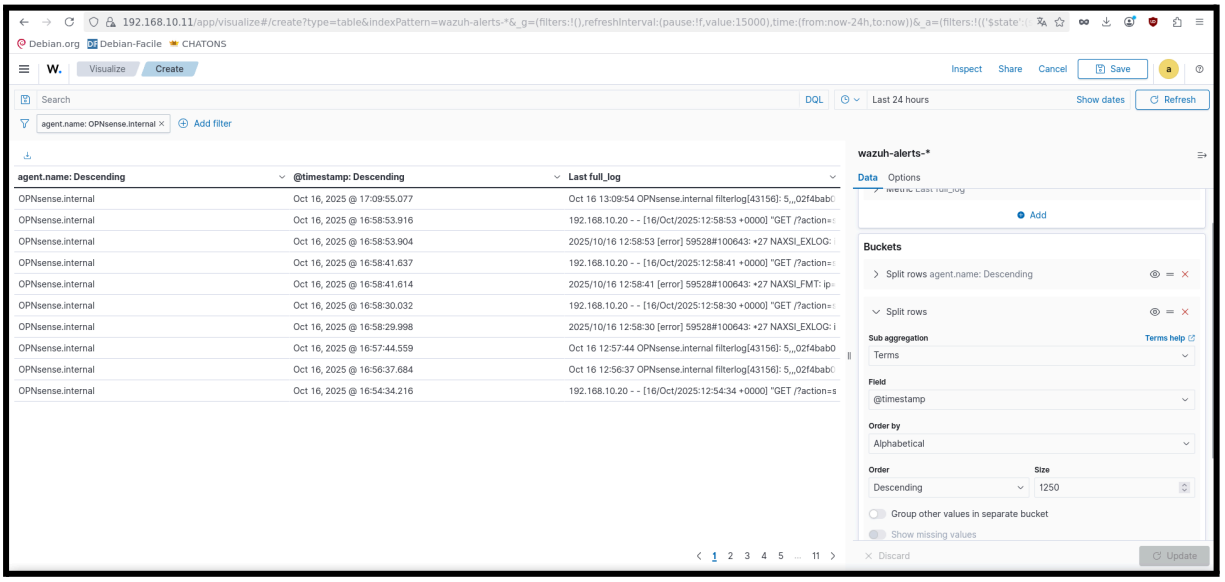
Chaque panneau du dashboard affichera ainsi les logs d'un agent spécifique, par exemple OPNsense dans le cas suivant :



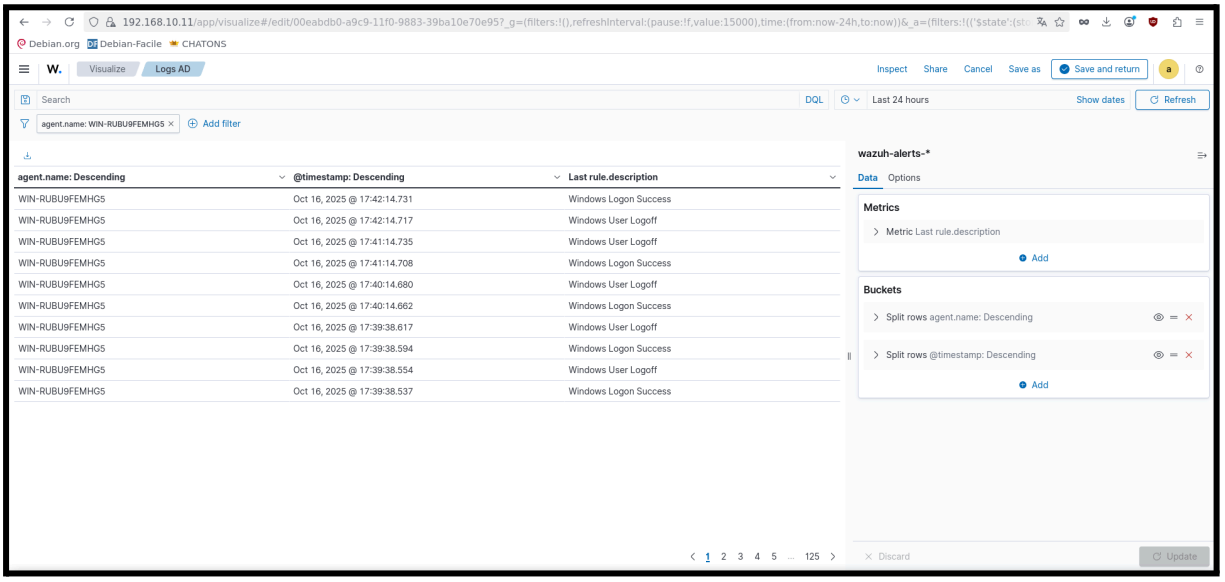

Ensuite, dans "Buckets" > "Split rows", nous sélectionnons "Terms" comme type d'agrégation, puis renseignons agent.name dans Field et fixons la taille à 1250. Nous créons deux buckets : l'un basé sur agent.name et l'autre sur @timestamp :



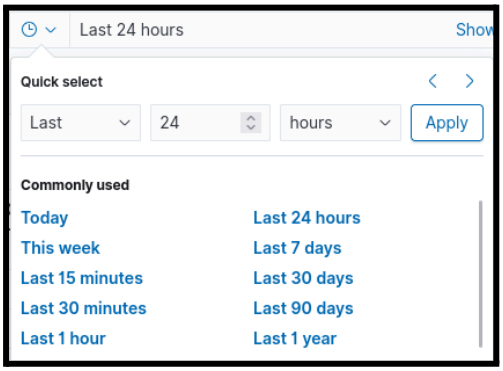
Puis en rafraichissant la page du dashboard, nous avons maintenant le panneau des logs OPNsense :

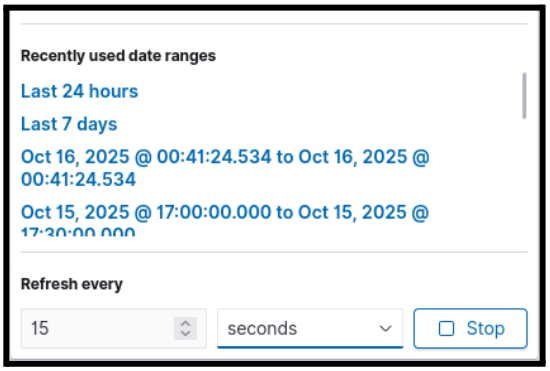



Idem pour le serveur Linux mais pour l'Active Directory, c'est un peu différent car les logs sont stockés dans des champs JSON spécifiques (comme win.eventdata, win.system...) et non dans full_log. Nous pouvons donc choisir d'afficher tous les champs ou uniquement la description du log via le champ rule.description :

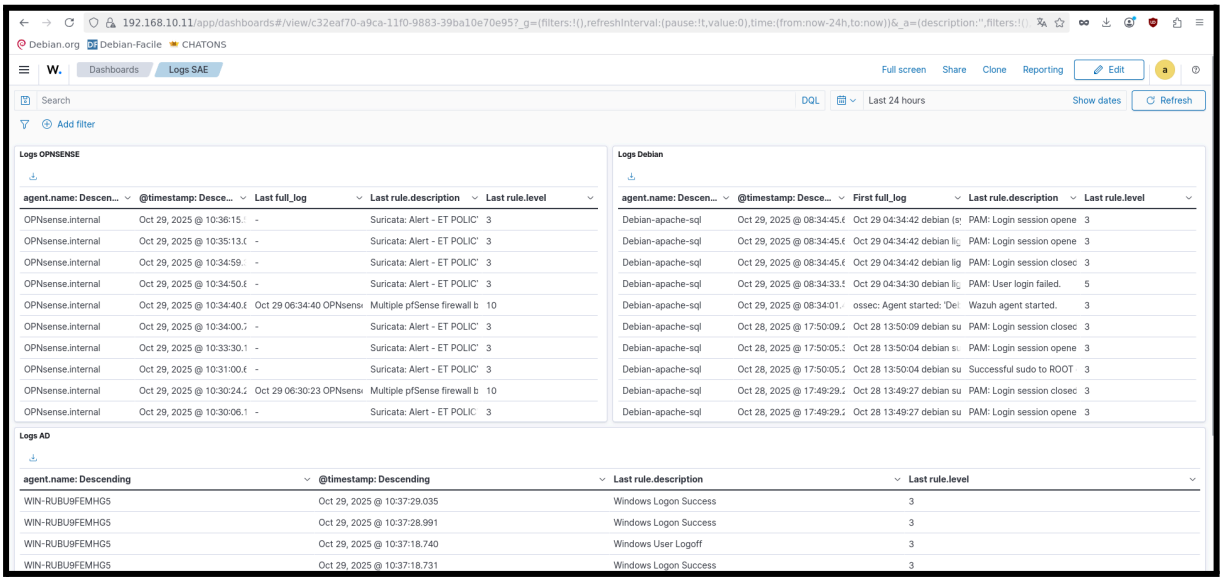


Une fois les trois panneaux configurés, nous activons l'actualisation automatique du dashboard, c'est la petite touche finale qui rend les dashboards plus intuitifs et simple d'usage :




Et nous avons donc les trois dashboards des logs de chacun des équipements du réseau :



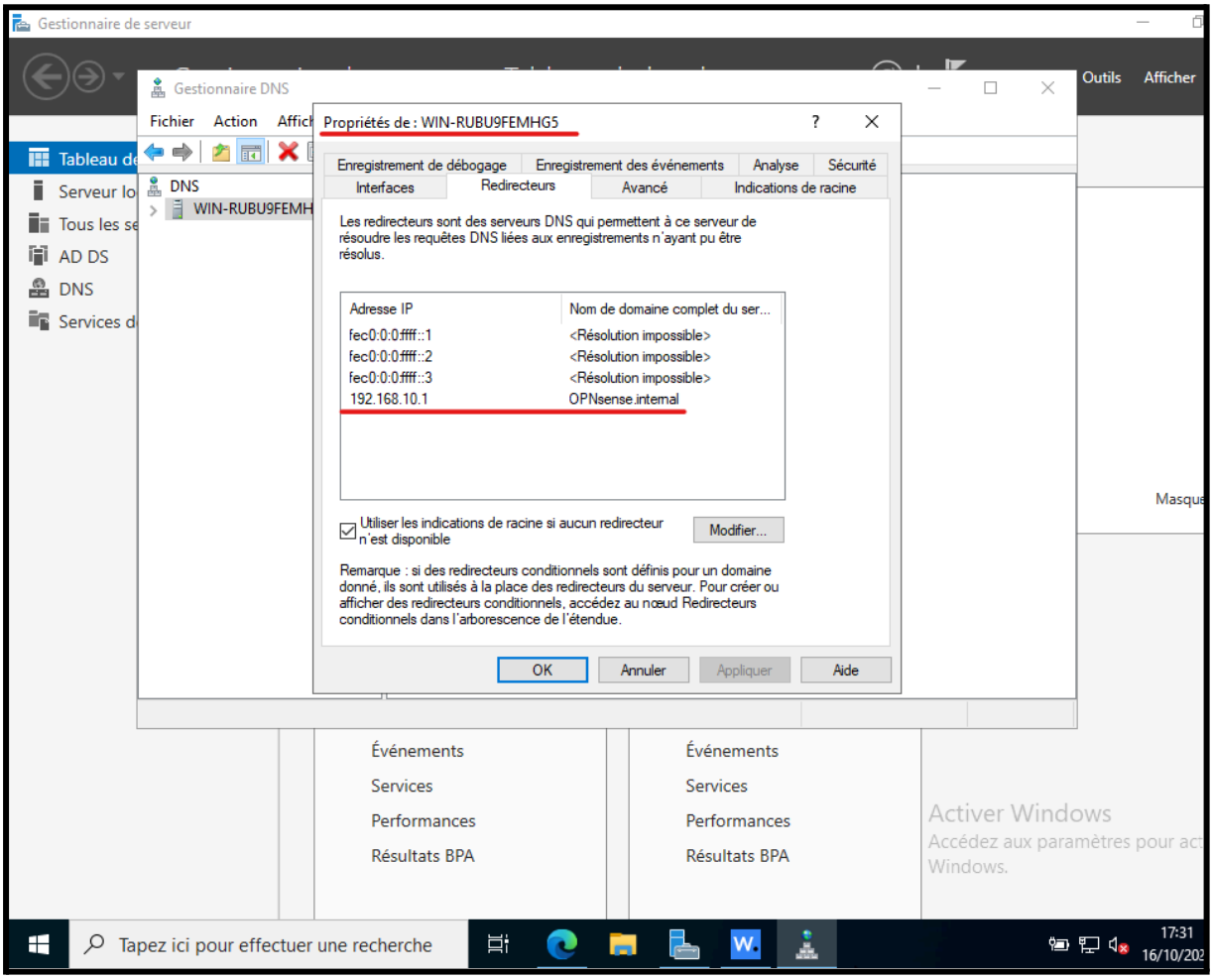

A noter que nous avons ajouté un champ rule.level permettant d'indiquer le niveau de criticité du log reçu.

Nous avons donc mis en place un dashboard personnalisé permettant de visualiser en temps réel les logs collectés par tous les agents Wazuh. Chaque panneau affiche les événements d'un équipement spécifique, avec un filtrage adapté et une actualisation automatique pour un suivi continu de l'activité du réseau.

## 5 - Configuration d'Active Directory

Pour que le serveur Active Directory puisse résoudre les noms de domaine et accéder à Internet, nous avons simplement modifié les paramètres DNS en ajoutant les serveurs nécessaires.

Cette étape garantit que le serveur AD pourra communiquer correctement avec le réseau et accéder aux mises à jour ou autres services externes si besoin :

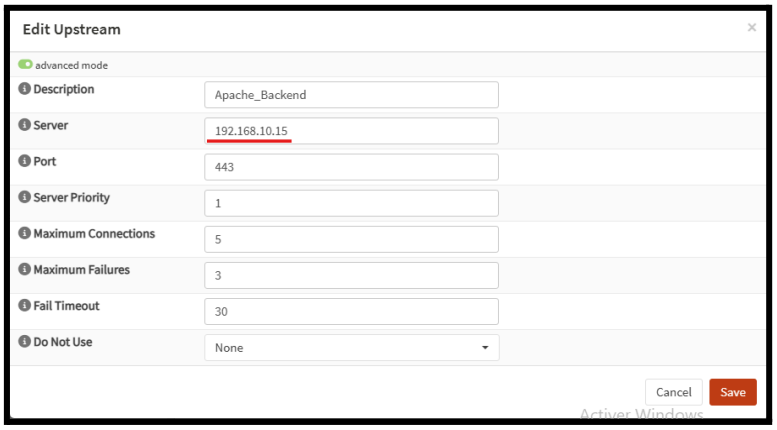



Après avoir ajouté OPNsense comme redirecteur DNS, le serveur Active Directory peut désormais résoudre correctement les noms de domaine externes et communiquer avec Internet. Cette configuration assure le bon fonctionnement des services AD et prépare le terrain pour la supervision et l'échange de logs avec Wazuh.

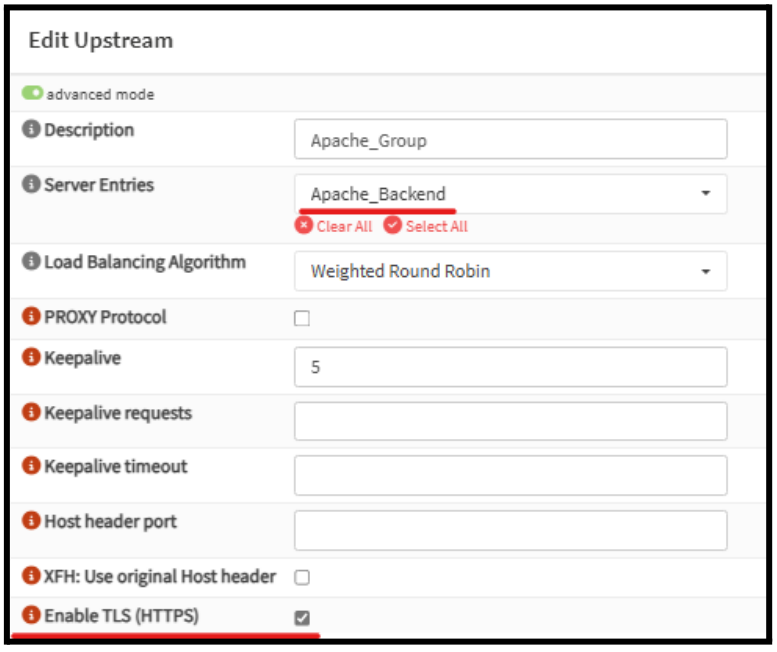
## 6 - Configuration du pare-feu OPNsense et du WAF

Dans cette partie, nous configurons OPNsense pour qu'il agisse à la fois comme pare-feu, reverse proxy et WAF (Web Application Firewall). L'objectif est de rediriger le trafic web sécurisé (HTTPS) vers le serveur Apache tout en filtrant les requêtes malveillantes grâce aux règles de sécurité du module Nginx.

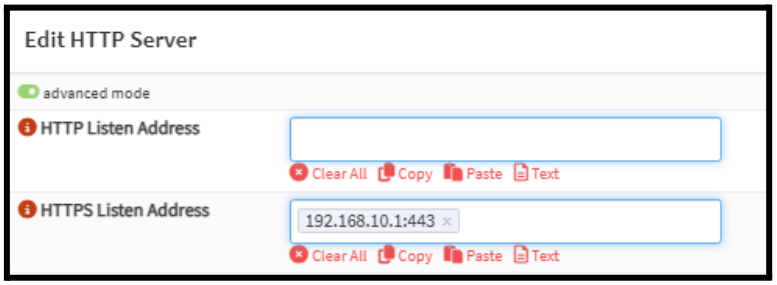
Nous commençons par ajouter notre serveur Apache en tant que upstream server, afin qu'OPNsense puisse lui rediriger les requêtes entrantes via le reverse proxy :



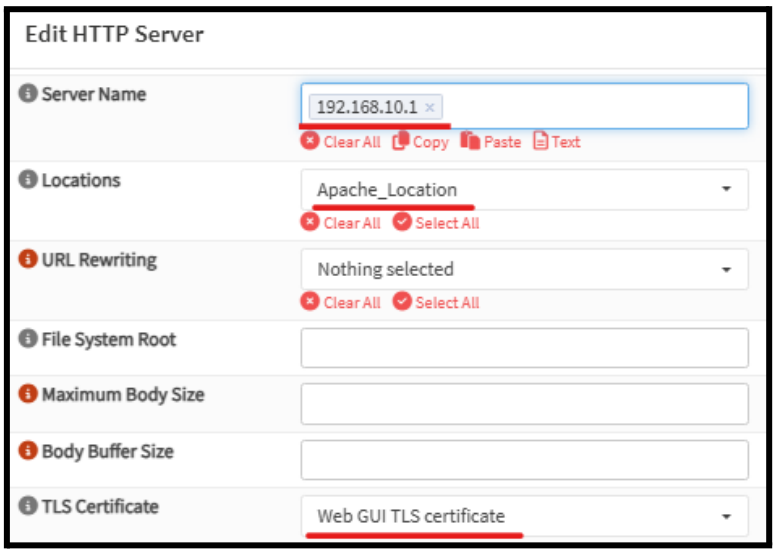

Toujours dans Upstream, nous sélectionnons l'upstream précédemment créé dans Server Entries et activons TLS pour le chiffrement HTTPS :



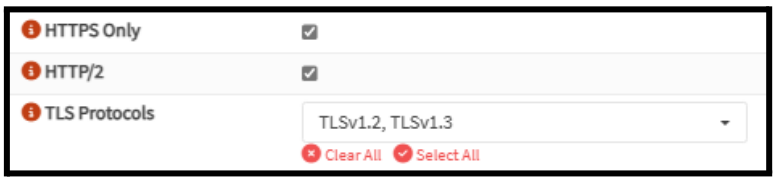
Dans HTTPS Server, nous renseignons l'adresse IP d'OPNsense et le port 443, afin qu'il écoute le trafic HTTPS et le redirige vers le serveur Apache :



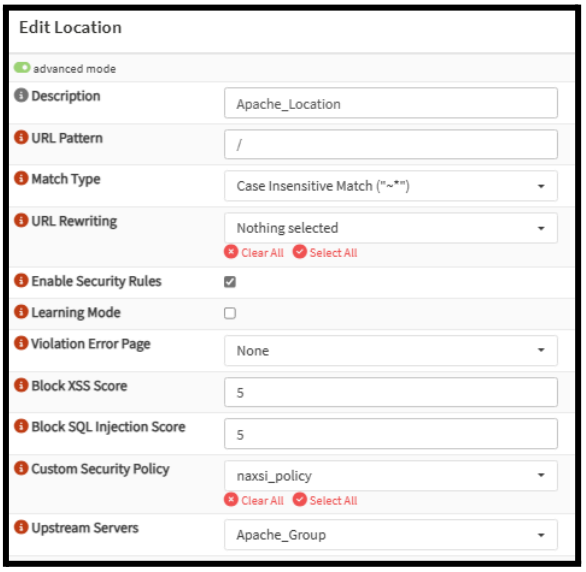

Le Server Name correspond à l'adresse d'OPNsense et le TLS Certificate permet de sécuriser la connexion HTTPS :



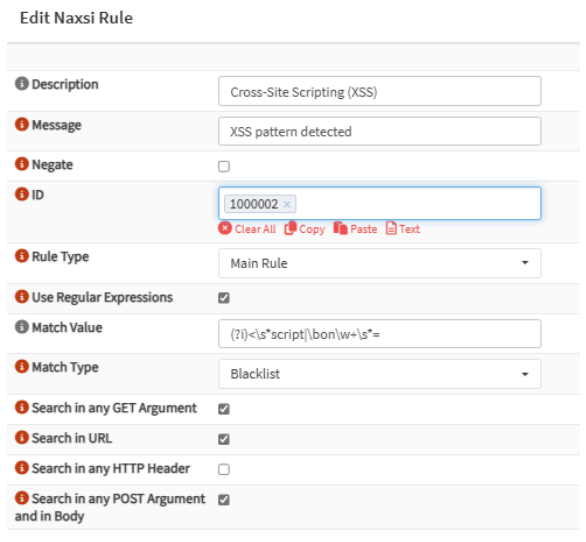
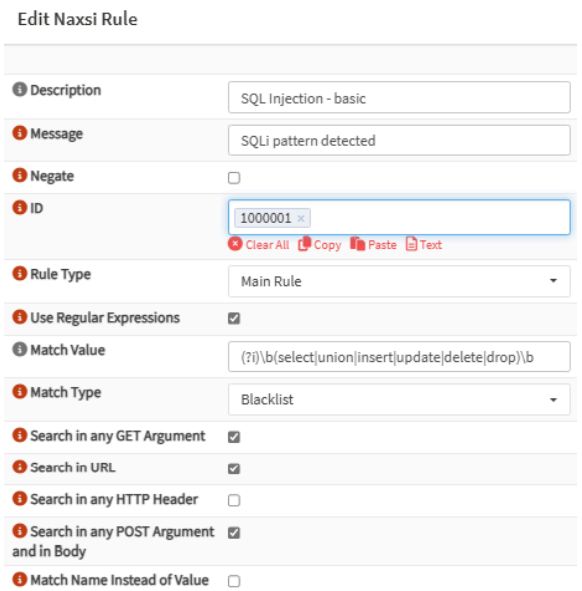

Et nous n'autorisons que les deux dernières versions TLS :



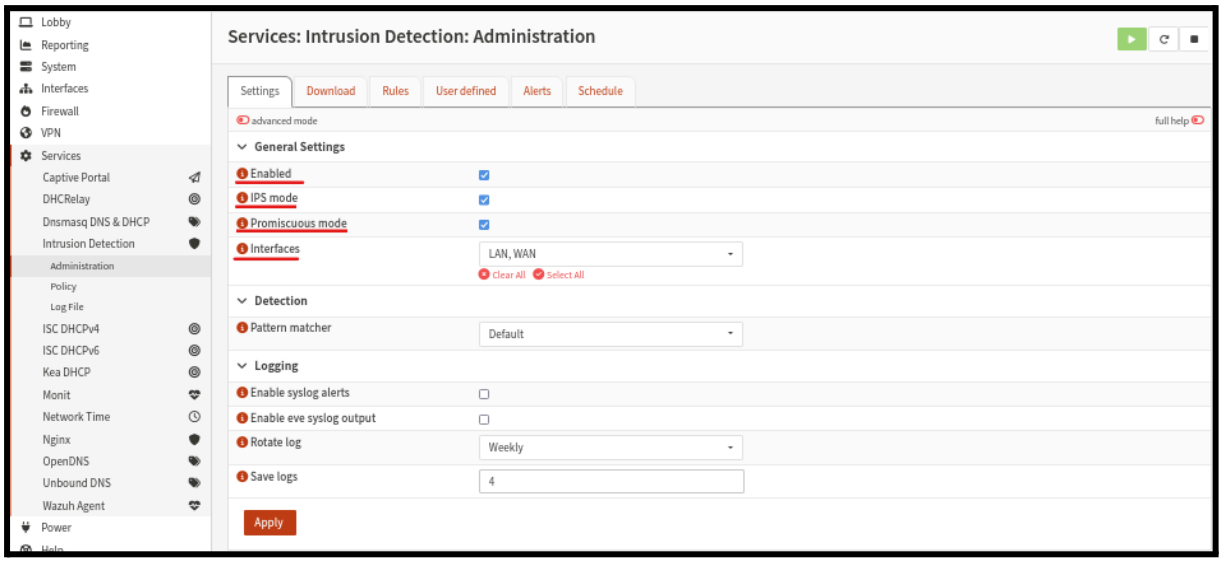
La section Location définit les règles du WAF. Les deux blocs permettent de bloquer les attaques, l'option Enable Security Rules active la protection WAF sur le reverse proxy, MatchType précise les éléments de l'URL à surveiller en cas d'injection, Upstream référence celui configuré précédemment, et Naxsi Policy correspond aux règles que créées par la suite :



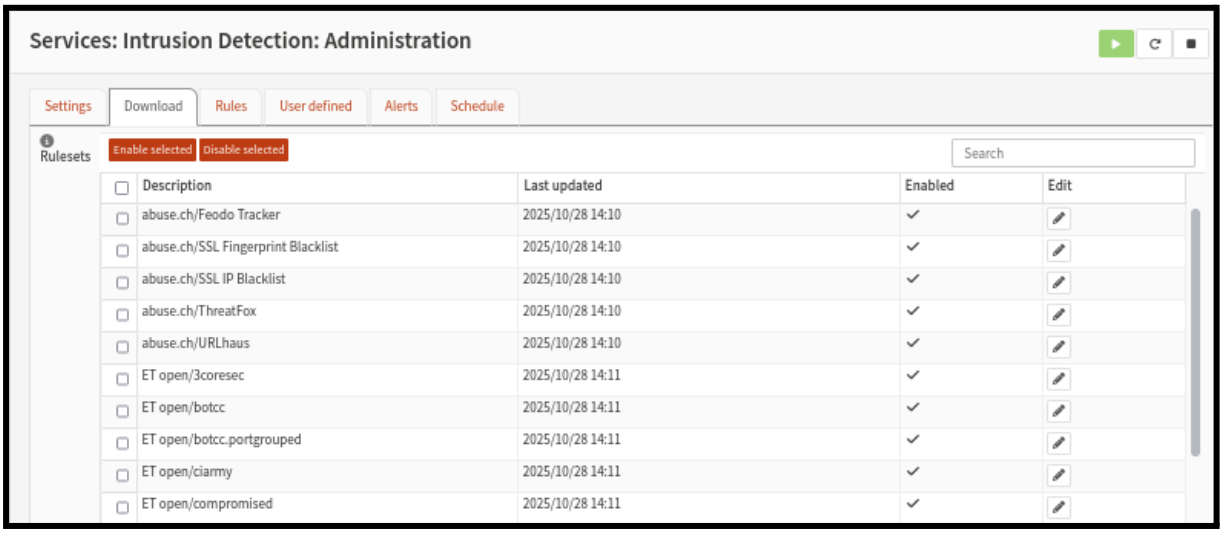

La première règle vise à bloquer les attaques XSS, le champ Match Value est essentiel car il définit le motif à identifier et à bloquer dans les requêtes. La deuxième règle est destinée à bloquer les tentatives d'injection, en filtrant les requêtes contenant des motifs suspects :

| Première Règle | Seconde Règle |
| --- | --- |
|  |  |

Nous activons également les systèmes de détection et de prévention d'intrusion (IDS et IPS) :

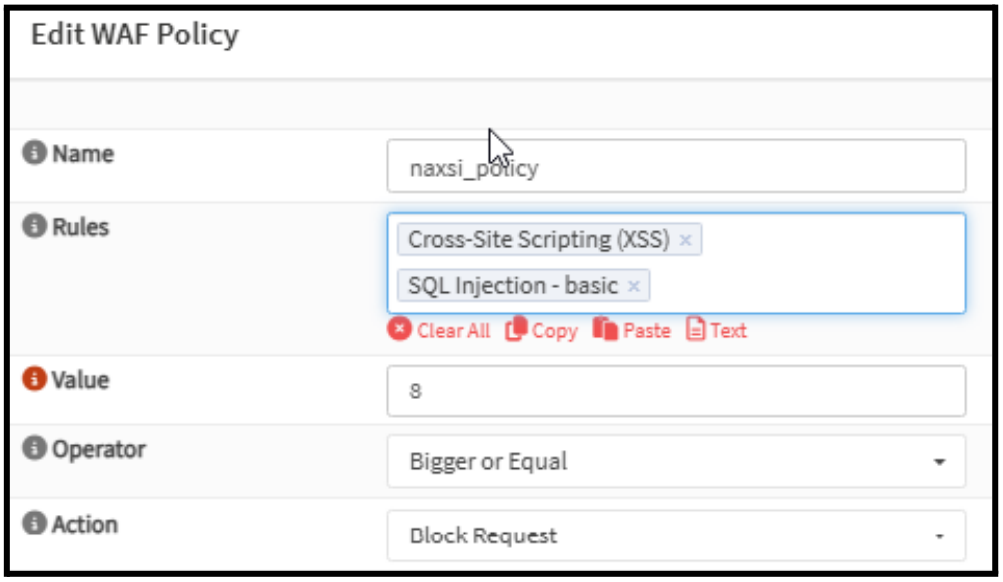



Puis nous autorisons toutes les règles de logging pertinentes afin de bien conserver des traces des-dites tentatives d'intrusion :




Et le dashboard peut à présent afficher les alertes liées au tentatives d'intrusion (une reconnaissance Nmap dans le cas ci-dessous) :

Nous créons ensuite une politique WAF que nous plaçons dans la section Location, elle regroupe et applique nos deux règles de sécurité :



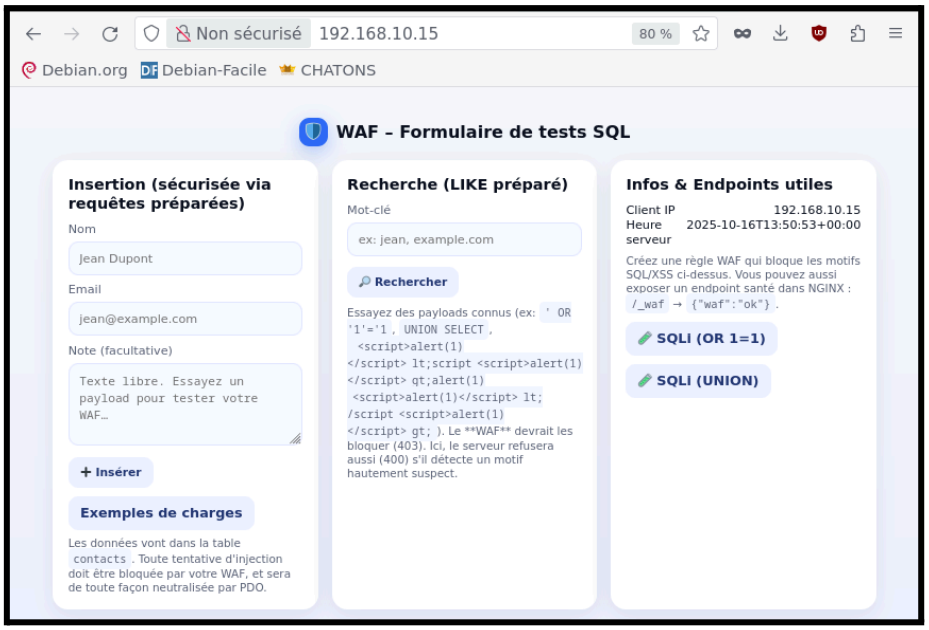

Étant donné que nous utilisons Nginx, nous ajoutons ses logs à la supervision Wazuh afin de centraliser la collecte des événements de sécurité :

```
<!-- NGINX / NAXSI logs -->
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/nginx/192.168.10.1.error.log</location>
  </localfile>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/nginx/waf_denied.access.log</location>
  </localfile>

  <!-- Active response -->
  <active-response>
    <disabled>no</disabled>
  </active-response>
```

Nous pouvons alors redémarrer nginx sur l'OPNsense au moyen de la commande suivante :

```
service nginx restart
```

Le serveur Apache est accessible à l'adresse 192.168.10.15, et voici la page web qu'il affiche :



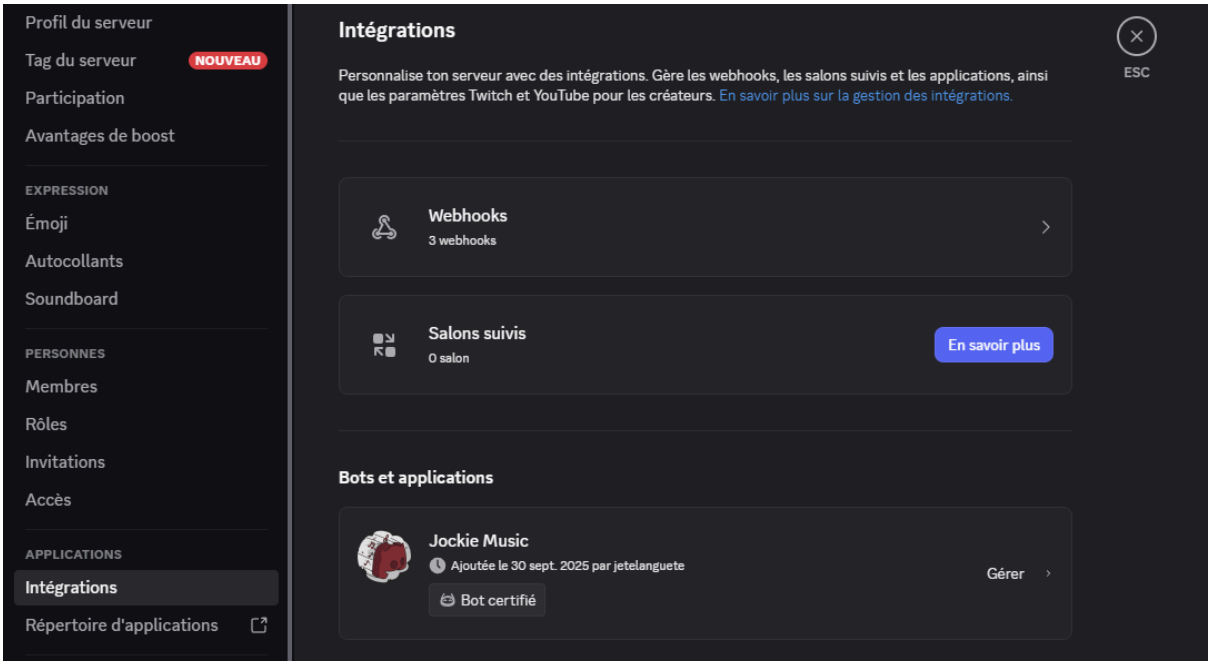

En accédant à 192.168.10.1 (l'adresse de notre WAF), la page affichée est la même que celle du serveur Apache, grâce à la redirection configurée via le reverse proxy :


A noter comme différence est que le site accédé depuis l'adresse du WAF est sécurisé et certifié avec un certificat auto-signé. Tout utilisateur accédant depuis l'extérieur ne voit que la version filtrée et sécurisée par le WAF, il n'accède jamais directement au serveur Apache. Ainsi, OPNsense assure à la fois la redirection du trafic, la protection contre les attaques web et la centralisation des logs pour Wazuh, garantissant une supervision et une sécurité renforcées du serveur web.

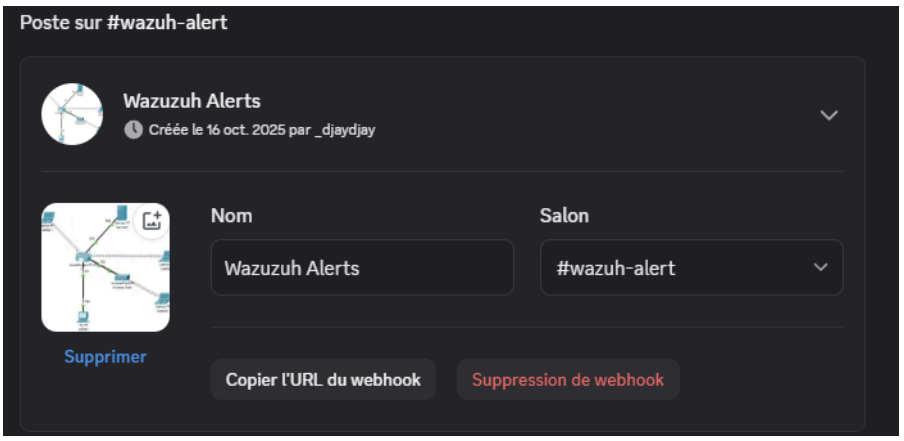
## 7 - BONUS : Intégration des alertes Wazuh sur Discord

Bien que non obligatoire, nous décidons d'intégrer le système d'alerte Wazuh sur Discord, cela nous permettra de nous entraîner pour la SAE et de malgré tout améliorer la réactivité de la surveillance. Cette intégration permet de recevoir en temps réel les notifications d'événements critiques sur le réseau, sans avoir à consulter l'interface web Wazuh.

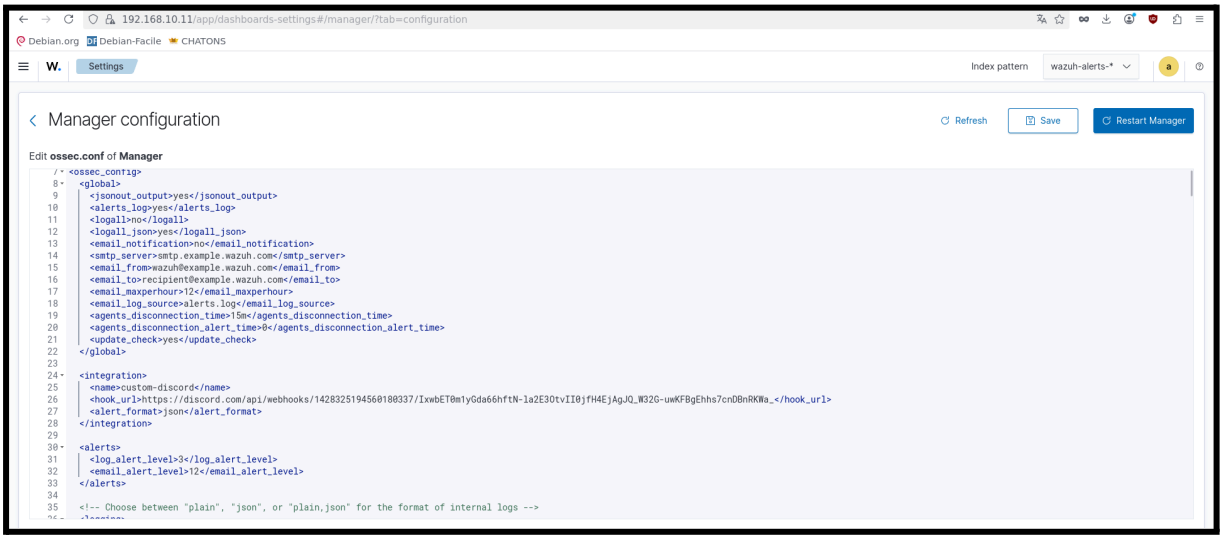
Pour cela, nous créons d'abord un webhook Discord associé au salon choisi :




Puis nous le nommons afin de pouvoir le reconnaître plus facilement et mettons de côté son URL qui nous servira plus tard :



Puis modifions le fichier ossec.conf du serveur Wazuh et ajoutons une balise <integration> afin configurer l'intégration des alertes Wazuh vers Discord :




Voici la balise <integration> que nous avons ajouté :

```
<integration>
    <name>custom-discord</name>
<hook_url>https://discord.com/api/webhooks/1428325194560180337/IxwbET0m1
y Gda66hftN-la2E3OtvII0jfH4EjAgJQ_W32G-uwKFBgEhhs7cnDBnRKWa_</hook_url>
    <alert_format>json</alert_format>
</integration>
```

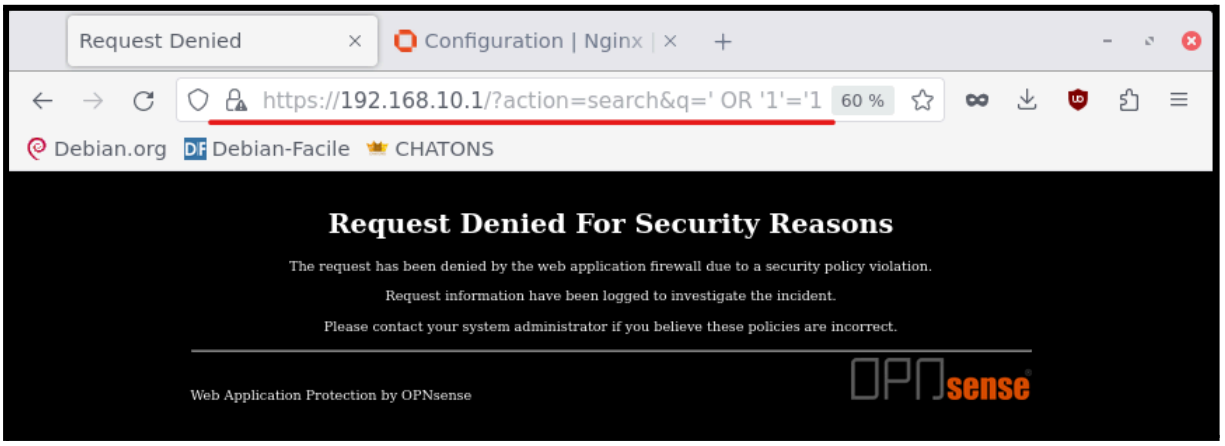
L'hook URL est l'URL du webhook Discord que nous venons de créer. Nous pouvons alors sauvegarder la configuration et redémarrer le service Wazuh et le tour est joué.

La démonstration du fonctionnement de ces alertes Discord sera faite dans la partie juste en dessous.

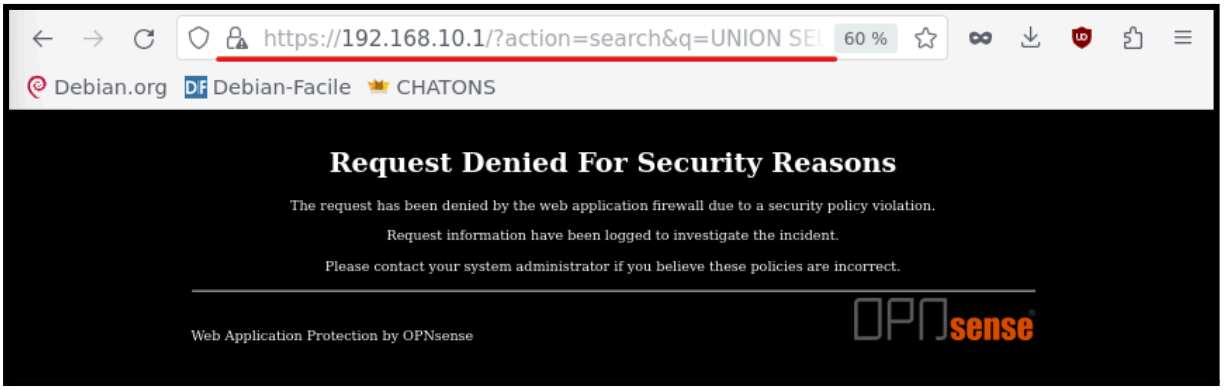
## 8 - Vérification du fonctionnement

Nous testons le dispositif en simulant une attaque d'injection sur l'application web protégée par le WAF, puis nous vérifions que l'attaque est bloquée localement, que les événements sont enregistrés dans les logs Nginx, qu'ils remontent dans Wazuh et enfin qu'une alerte est transmise sur Discord.

Commençons donc par effectuer une injection SQL dans l'URL en ajoutant une expression qui sera toujours vraie (1 = 1), dans l'espoir d'exfiltrer des données :




Le WAF joue correctement son rôle en bloquant la requête malveillante, nous pouvons en tenter une différente qui permet toujours d'exfiltrer des données (dans les numéros de colonnes spécifiés) :



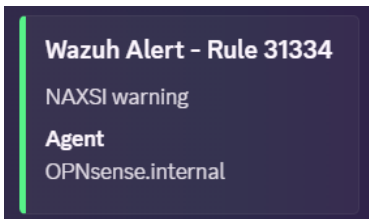

Une requête malveillante qui est à nouveau bloquée par le WAF.

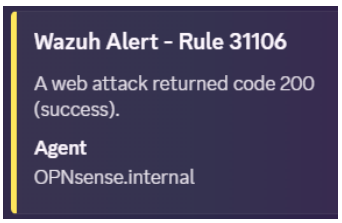
Nous consultons les logs Nginx d'OPNsense pour confirmer l'entrée du blocage et relever les détails de la requête bloquée :

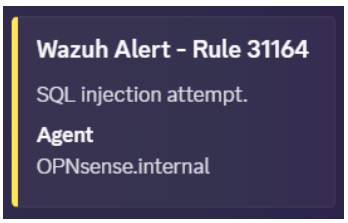
```
192.168.10.15 - - [16/Oct/2025:13:53:08 +0000] "GET /?action=search&q=%27%20OR%20%271%27=%271 HTTP/2.0"
 200 5232 "https://192.168.10.1/" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115
.0" "-"
192.168.10.15 - - [16/Oct/2025:13:54:19 +0000] "GET /?action=search&q=UNION%20SELECT%201,2,3 HTTP/2.0"
200 5232 "https://192.168.10.1/" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.
0" "-"
```

```
2025/10/16 13:54:19 [error] 59528#100643: *31 NAXSI_EXLOG: ip=192.168.10.15&server=192.168.10.1&rid=740
4d84dcadb50b83f1611d7925ebadd&uri=%2F&id=17&zone=ARGS&var_name=q&content=UNION%20SELECT%201%2C2%2C3, cl
ient: 192.168.10.15, server: 192.168.10.1, request: "GET /?action=search&q=UNION%20SELECT%201,2,3 HTTP/
2.0", host: "192.168.10.1", referrer: "https://192.168.10.1/"
2025/10/16 13:54:19 [error] 59528#100643: *31 NAXSI_FMT: ip=192.168.10.15&server=192.168.10.1&uri=/&con
fig=block&rid=7404d84dcadb50b83f1611d7925ebadd&cscore0=$LIBINJECTION_SQL&score0=8&zone0=ARGS&id0=17&var
_name0=q, client: 192.168.10.15, server: 192.168.10.1, request: "GET /?action=search&q=UNION%20SELECT%2
01,2,3 HTTP/2.0", host: "192.168.10.1", referrer: "https://192.168.10.1/"
```

Et voici les alertes que nous recevons sur Discord :




Les injections testées (OR '1'='1', UNION SELECT ...) sont systématiquement bloquées par le WAF, empêchant toute exfiltration ou exécution non désirée côté application. Les blocages génèrent des entrées claires dans les logs Nginx d'OPNsense (URI, règle WAF, horodatage) qui sont ensuite collectées et indexées par Wazuh, assurant la traçabilité de l'événement. Enfin, Wazuh transmet automatiquement les alertes au salon Discord via le webhook, offrant une visibilité immédiate.

La mise en place de la supervision centralisée avec Wazuh a permis de collecter, corréler et analyser les logs de tous les équipements du réseau, offrant une visibilité complète sur l'activité du système. La configuration du WAF sur OPNsense a assuré la protection des services web, bloquant les injections malveillantes et garantissant que les utilisateurs externes n'accèdent qu'à la version filtrée et sécurisée des applications. Les logs des blocages sont centralisés dans Wazuh, assurant une traçabilité complète et un suivi des incidents.

Ce projet illustre l'importance d'une supervision intégrée et proactive pour renforcer la sécurité des systèmes d'information, tout en fournissant des outils pratiques pour l'analyse et la réaction face aux menaces.

## 9 - Bibliographie

Guide pour l'installation de Wazuh :

https://it-admin.tg/installer-wazuh-sous-linux-debianubuntumint/

Guide pour l'intégration des alertes Wazuh sur Discord

https://www.learntohomelab.com/homelabseries/EP19_wazuhdiscordalerts/

Guide pour la collecte complète des logs Wazuh, y compris ceux non affichés par défaut :

https://documentation.wazuh.com/current/user-manual/manager/event-logging.html#archiving-event-logs

Guide pour le rafraîchissement automatique du dashboard Wazuh des logs des équipements :

https://www.reddit.com/media?url=https%3A%2F%2Fpreview.redd.it%2Fauto-refreshing-dashboard-v0-p9r8yonmjlmb1.png%3Fwidth%3D609%26auto%3Dwebp%26s%3D2a661724c36648b0a8c42bca88a5bd25be5db062