
**RAPPORT TP Wazuh SAÉ 5.CYBER.03**

**Université de la réunion / IUT**

# Département Réseaux, Télécommunication en Cybersécurité – 3ème années

**SAÉ 5.CYBER.03**

**ASSURER LA SÉCURISATION ET LA SUPERVISION AVANCÉES D'UN SYSTÈME**


# EMMANUEL GRONDIN

# Yann BOISVILLIERS

# Compte Rendu Complet d'Installation de Wazuh SIEM

## 1. Introduction

Ce rapport présente la mise en œuvre complète d'une infrastructure de supervision et de sécurité avancée basée sur Wazuh SIEM, dans le cadre de la SAÉ 5.CYBER.03 portant sur la sécurisation et la supervision avancées d'un système.

Face aux menaces croissantes pesant sur les infrastructures informatiques, il est devenu primordial de disposer d'une solution centralisée capable de détecter, analyser et réagir aux incidents de sécurité en temps réel. Wazuh, en tant que plateforme SIEM (Security Information and Event Management) open-source, répond à ces exigences en offrant une visibilité complète sur l'ensemble des événements de sécurité d'une infrastructure.

Ce travail s'articule autour de trois axes principaux :

- **Le déploiement technique** d'une architecture Wazuh complète et sécurisée, comprenant l'indexeur OpenSearch, le gestionnaire Wazuh Manager et le tableau de bord Dashboard, le tout dans un environnement virtuel sous Debian
- **L'intégration multi-sources** avec la supervision de différents systèmes critiques : serveur web Apache, pare-feu OPNsense avec son IDS/IPS Suricata et WAF NAXSI, ainsi qu'un contrôleur de domaine Active Directory
- **L'automatisation et l'amélioration de la réactivité** via une intégration avec Discord permettant une notification instantanée des alertes de sécurité

Cette approche holistique permet non seulement de centraliser les journaux de sécurité provenant de sources hétérogènes, mais également de détecter et bloquer proactivement les tentatives d'intrusion, qu'il s'agisse d'injections SQL, d'attaques XSS ou d'autres vecteurs d'attaque couramment exploités.

## 2. Configuration du Système

**Architecture déployée :** Installation centralisée sur un serveur unique
**Système d'exploitation :** Debian (Machine Virtuelle)
**Adresse IP du serveur :** 192.168.1.30
**Mode réseau :** Mode pont
**Version de Wazuh :** 4.7

# 3. Préparation du Système

## Contexte et objectifs

Avant d'installer Wazuh, il est indispensable de préparer l'environnement système pour garantir l'authenticité et l'intégrité des paquets téléchargés. Cette étape établit une relation de confiance avec les dépôts officiels Wazuh.

## 3.1 Importation de la clé GPG Wazuh

La clé GPG (GNU Privacy Guard) permet de vérifier que les paquets Wazuh n'ont pas été altérés ou compromis durant leur téléchargement. C'est une mesure de sécurité fondamentale.

```
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
```

**Décomposition de la commande :**

- curl -s : Télécharge silencieusement la clé publique depuis le serveur Wazuh
- gpg --import : Importe la clé dans un trousseau GPG spécifique
- --no-default-keyring : Évite d'utiliser le trousseau par défaut de l'utilisateur
- --keyring gnupg-ring. : Crée un trousseau dédié pour Wazuh
- chmod 644 : Définit les permissions en lecture pour tous, mais écriture uniquement pour root

## 3.2 Ajout du dépôt Wazuh

Cette configuration indique au gestionnaire de paquets APT où trouver les paquets Wazuh officiels et quelle clé utiliser pour vérifier leur authenticité.

```
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
```

**Éléments clés :**

- deb : Type de dépôt (paquets binaires Debian)
- signed-by= : Spécifie la clé GPG pour valider les paquets
- stable main : Canal de distribution et composant principal

### 3.3 Mise à jour et installation des dépendances

Ces paquets sont nécessaires au bon fonctionnement de Wazuh. Chacun a un rôle spécifique dans l'architecture.

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

# 4. Configuration des Certificats SSL/TLS

## Contexte et importance

Les certificats SSL/TLS constituent la pierre angulaire de la sécurité dans Wazuh. Ils garantissent :

- **Chiffrement** : Protection des données en transit
- **Authentification** : Vérification de l'identité des composants
- **Intégrité** : Assurance que les données n'ont pas été modifiées

Dans une architecture Wazuh, **tous** les composants communiquent via HTTPS/TLS :

- Filebeat → Indexer (port 9200)
- Dashboard → Indexer (port 9200)
- Manager ↔ Agents (port 1514/1515)

## 4.1 Téléchargement des outils de génération

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

**Fichiers téléchargés :**

- wazuh-certs-tool.sh : Script automatisé de génération des certificats
- config.yml : Fichier de définition de l'architecture (nœuds et IPs)

## 4.2 Modification du fichier config.yml

**Pourquoi personnaliser ce fichier ?** Le fichier `config.yml` définit **l'architecture complète** de votre déploiement Wazuh. Chaque nœud déclaré recevra son propre certificat avec un CN (Common Name) unique.

**Configuration appliquée pour une installation centralisée :**

```
nodes:
  # Wazuh indexer nodes
  indexer:
```

```
  - name: node-1
    ip: "192.168.1.30"
  #- 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: "192.168.1.30"
  #  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: "192.168.1.30"
```

**Explication des choix architecturaux :**

- **Un seul serveur** (192.168.1.30) héberge tous les composants
- **Noms explicites** : node-1, wazuh-1, dashboard
- **Architecture simplifiée** : Idéale pour PoC, lab, ou PME
- **Évolutivité** : Lignes commentées montrent comment ajouter des nœuds additionnels

**Scénario multi-nœuds (commenté) :** Dans un environnement de production, vous pourriez décommenter et configurer :

- Plusieurs indexers pour la haute disponibilité
- Des workers Wazuh pour la répartition de charge
- Des dashboards multiples pour la tolérance aux pannes

### 4.3 Génération des certificats

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

**Certificats générés :**

- Admin (admin.pem, admin-key.pem)
- Indexer (node-1.pem, node-1-key.pem)
- Dashboard (dashboard.pem, dashboard-key.pem)
- Serveur (wazuh-1.pem, wazuh-1-key.pem)
- Root CA (root-ca.pem, root-ca.key)

# 5. Installation des Paquets Wazuh

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

---

# 6. Configuration de l'Indexer Wazuh

## Contexte et rôle

L'**Indexer Wazuh** (basé sur OpenSearch) est le cœur du système SIEM. Il remplit trois fonctions critiques :

1. **Stockage** : Conservation des événements de sécurité
2. **Indexation** : Organisation des données pour recherche rapide
3. **Recherche** : Requêtes analytiques en temps réel

C'est l'équivalent d'une base de données spécialisée pour les logs de sécurité.

## 6.1 Édition du fichier de configuration

```
nano /etc/wazuh-indexer/opensearch.yml
```

**Paramètres principaux configurés :**

```
network.host: "192.168.1.30"
node.name: "node-1"
cluster.initial_master_nodes:
- "node-1"
#- "node-2"
#- "node-3"
cluster.name: "wazuh-cluster"
#discovery.seed_hosts:
#  - "node-1-ip"
#  - "node-2-ip"
#  - "node-3-ip"
node.max_local_storage_nodes: "3"
path.data: /var/lib/wazuh-indexer
```

```
path.logs: /var/log/wazuh-indexer

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

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

plugins.security.system_indices.enabled: true
plugins.security.system_indices.indices: [".plugins-ml-model", ".plugins-ml-task",
".opendistro-alerting-config", ".opendistro-alerting-alert*", ".opendistro-anomaly-results*",
".opendistro-anomaly-detector*", ".opendistro-anomaly-checkpoints",
".opendistro-anomaly-detection-state", ".opendistro-reports-*", ".opensearch-notifications-*",
".opensearch-notebooks", ".opensearch-observability",
".opendistro-asynchronous-search-response*", ".replication-metadata-store"]

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

**Paramètres réseau :**

- network.host: IP d'écoute (192.168.1.30 ou 0.0.0.0 si multi-interface, pas 127.0.0.1)
- node.name: nom unique du nœud (node-1)

**Cluster :**

- cluster.name: nom du cluster (wazuh-cluster)
- cluster.initial_master_nodes: liste des nœuds maîtres (ex. node-1)

**Stockage :**

- path.data: /var/lib/wazuh-indexer
- path.logs: /var/log/wazuh-indexer
  → Disque séparé conseillé, alerte à 85 %, rotation des logs.

**SSL/TLS HTTP (API REST, port 9200) :**

- Certificats (indexer.pem, indexer-key.pem, root-ca.pem)
- enabled: true → HTTPS activé

**SSL/TLS Transport (inter-nœuds, port 9300) :**

- Même certificats
- enforce_hostname_verification: false, resolve_hostname: false
  → OK en lab, à sécuriser en production.

**Certificats administratifs :**

- admin_dn: DN du certificat admin (plein accès)
- nodes_dn: DN des nœuds autorisés (contrôle d'accès au cluster)

**Compatibilité Filebeat :**

- compatibility.override_main_response_version: true
  → Assure la compatibilité avec Filebeat 7.10.2.

---

## 6.2 Déploiement des certificats de l'indexer

```
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
```

**Explication étape par étape :**

1. **Extraction ciblée** :
   - Extrait uniquement les certificats nécessaires pour ce nœud
   - Évite de copier tous les certificats inutilement

2. **Renommage générique** :
   - `node-1.pem` → `indexer.pem`
   - Uniformise les noms pour correspondre aux chemins dans opensearch.yml
3. **Sécurisation des permissions** :
   - `500` sur le répertoire : Lecture + exécution uniquement pour le propriétaire
   - `400` sur les fichiers : Lecture seule pour le propriétaire
   - Protection contre la lecture par d'autres utilisateurs
4. **Attribution de propriété** :
   - Le processus `wazuh-indexer` doit pouvoir lire ses propres certificats

---

### 6.3 Activation et démarrage du service

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

### 6.4 Initialisation du cluster

```
/usr/share/wazuh-indexer/bin/indexer-security-init.sh
```

`indexer-security-init.sh` : Configure les utilisateurs, rôles et politiques de sécurité

---

## 7. Configuration du Wazuh Manager

### 7.1 Activation et démarrage du service

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

---

## 8. Configuration de Filebeat

Filebeat est un **agent de collecte de logs** développé par **Elastic**.
Fonction : lire, traiter et envoyer des fichiers journaux vers une destination comme **Elasticsearch**, **OpenSearch** ou **Logstash**.

Utilité dans Wazuh :

- Transfère les **alertes de sécurité** générées par Wazuh vers **Wazuh Indexer (OpenSearch)**.
- Permet leur **indexation, recherche et visualisation** dans le **Wazuh Dashboard**.

## 8.1 Installation du paquet Filebeat

```
apt install filebeat
```

**Version requise :** Filebeat 7.10.2 (OSS) pour compatibilité avec OpenSearch

## 8.2 Téléchargement du fichier de configuration

```
curl -so /etc/filebeat/filebeat.yml https://packages.wazuh.com/4.7/tpl/wazuh/filebeat/filebeat.yml
```

**Pourquoi utiliser le template officiel ?**

- Configuration pré-optimisée pour Wazuh
- Modules et pipelines déjà définis
- Évite les erreurs de configuration courantes

## 8.3 Édition du fichier de configuration

```
nano /etc/filebeat/filebeat.yml
```

**Configuration appliquée :**

```
output.elasticsearch:
  hosts: ["192.168.1.30:9200"]
  protocol: https
  username: "admin"
  password: "admin"
  ssl.certificate_authorities:
    - /etc/filebeat/certs/root-ca.pem
  ssl.certificate: "/etc/filebeat/certs/filebeat.pem"
  ssl.key: "/etc/filebeat/certs/filebeat-key.pem"

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

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

La configuration de **Filebeat** définit la sortie vers **OpenSearch** en HTTPS avec authentification et certificats TLS.
Il faut utiliser l'adresse IP réelle de l'indexer et créer un compte utilisateur dédié en production.
Le **module Wazuh** est activé pour envoyer uniquement les alertes de sécurité, sans les archives brutes.
Les paramètres de **logs** définissent un niveau d'information standard, une rotation automatique et un stockage local dans `/var/log/filebeat`.
Objectif : assurer une transmission sécurisée, fiable et optimisée des alertes Wazuh vers OpenSearch.

**Note importante :** Utiliser l'adresse IP 192.168.1.30 et non 127.0.0.1 pour que Filebeat se connecte correctement à OpenSearch.

## 8.4 Création du Keystore

```
filebeat keystore create
```

**Pourquoi utiliser un keystore ?**

- Chiffre les identifiants au lieu de les stocker en clair
- Protection contre la lecture directe du fichier de configuration
- Bonne pratique de sécurité recommandée par Elastic

## 8.5 Ajout des identifiants

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

## 8.6 Téléchargement du modèle d'alerte

```
curl -so /etc/filebeat/wazuh-template.json
https://raw.githubusercontent.com/wazuh/wazuh/v4.7.2/extensions/elasticsearch/7.x/wazuh-tem
plate.json
chmod go+r /etc/filebeat/wazuh-template.json
```

**Rôle du template :**

- Définit le mapping des champs dans OpenSearch
- Optimise l'indexation et les recherches
- Définit les types de données (IP, date, text, keyword...)

### 8.7 Déploiement des 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
```

**Particularité :** Propriétaire `root:root` (vs `wazuh-indexer:wazuh-indexer` pour l'indexer)

### 8.8 Activation et démarrage du service

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

---

# 9. Configuration du Wazuh Dashboard

### Contexte et rôle :

Le **Wazuh Dashboard** est l'interface web de visualisation et d'analyse. Basé sur OpenSearch Dashboards (fork de Kibana), il offre :

- Tableaux de bord préconfigurés pour la sécurité
- Moteur de recherche avancé (Lucene)
- Visualisations personnalisables
- Gestion des agents Wazuh

### 9.1 Édition du fichier de configuration

```
nano /etc/wazuh-dashboard/opensearch_dashboards.yml
```

**Configuration appliquée :**

```
server.host: 192.168.1.30
server.port: 5601

opensearch.hosts: ["https://192.168.1.30:9200"]
opensearch.ssl.verificationMode: certificate
opensearch.username: "admin"
opensearch.password: "admin"
opensearch.ssl.certificateAuthorities: ["/etc/wazuh-dashboard/certs/root-ca.pem"]

server.ssl.enabled: true
server.ssl.key: "/etc/wazuh-dashboard/certs/dashboard-key.pem"
server.ssl.certificate: "/etc/wazuh-dashboard/certs/dashboard.pem"

uiSettings.overrides.defaultRoute: /app/wz-home
```

La configuration du **Wazuh Dashboard** définit l'adresse réseau (`192.168.1.30`) et le port d'accès (`5601`) pour l'interface web.
Le tableau de bord se connecte à **l'indexer OpenSearch** via HTTPS, avec vérification du certificat et authentification par identifiants.
Les certificats SSL assurent la sécurité des communications entre le Dashboard et l'indexer.
Le **HTTPS** est activé pour chiffrer les sessions utilisateurs et protéger les identifiants.
Enfin, la page d'accueil est configurée pour s'ouvrir directement sur l'interface principale de Wazuh.

## 9.2 Déploiement des certificats 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
```

**Particularité :** Propriétaire `wazuh-dashboard:wazuh-dashboard`

## 9.3 Activation et démarrage du service

```
systemctl daemon-reload
systemctl enable wazuh-dashboard
```

```
systemctl start wazuh-dashboard
```

# 10. Vérification des Services

## Objectif

Cette étape valide que tous les composants sont opérationnels et communiquent correctement.

## 10.1 Commandes de vérification

```
# Indexer
sudo systemctl status wazuh-indexer

# Manager
sudo systemctl status wazuh-manager

# Dashboard
sudo systemctl status wazuh-dashboard

# Filebeat
sudo systemctl status filebeat
```

## 10.2 Vérification des ports

```
sudo ss -tulnp | grep 9200   # Indexer
sudo ss -tulnp | grep 5601   # Dashboard
```

## 10.3 Test de la connexion Filebeat → Indexer

```
sudo filebeat test output
```

---

# 11. Accès à l'Interface Web

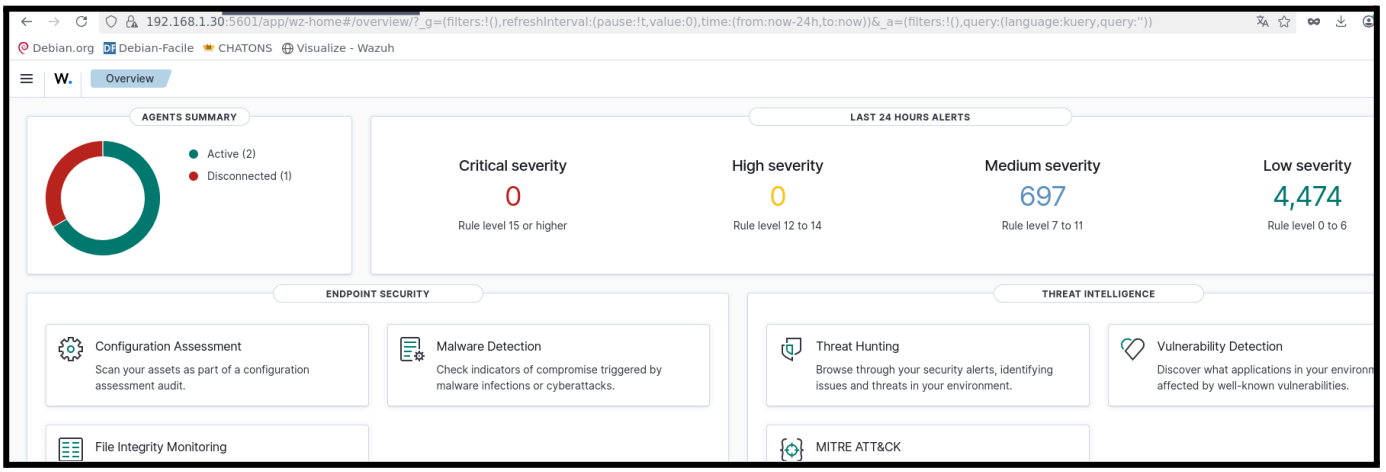
## Contexte

Après avoir configuré tous les composants, il est temps d'accéder à l'interface de gestion Wazuh pour vérifier le bon fonctionnement de l'installation.

**URL d'accès :** https://192.168.1.30:5601

**Identifiants par défaut :**

- **Utilisateur :** admin
- **Mot de passe :** admin




# 12. Résolution des Problèmes Courants

## 12.1 "Dashboard server is not ready yet"

**Solutions :**

- Vérifier que l'indexer est actif et accessible
- Vérifier que le Dashboard peut lire les certificats et se connecter à l'indexer
- Patienter 30-60 secondes après le démarrage des services

## 12.2 Problème Filebeat "connection refused"

**Solutions :**

- Vérifier l'IP et le port configurés (192.168.1.30:9200)
- Vérifier que l'indexer est bien actif
- Vérifier les certificats et la configuration SSL
- S'assurer d'utiliser l'adresse IP du serveur et non 127.0.0.1

# 13. État Final du Système

**Indexer OpenSearch** : Actif sur 192.168.1.30:9200
**Wazuh Manager** : Actif et opérationnel
**Filebeat** : Actif et envoie les alertes Wazuh vers l'indexer
**Wazuh Dashboard** : Accessible via https://192.168.1.30:5601
**Communications sécurisées** : Toutes les communications via SSL/TLS

# 14. Points Clés de Sécurité

- Tous les composants utilisent des certificats SSL/TLS pour les communications chiffrées
- Les permissions sur les certificats sont correctement configurées (500 pour les répertoires, 400 pour les fichiers)
- L'authentification est activée avec les identifiants admin par défaut (à modifier en production)
- La vérification du hostname est désactivée pour simplifier la configuration en environnement de test

---

# Supervision des logs Apache, OPNsens et Active directory avec Wazuh :

# Objectif :

L'objectif de cette partie est de surveiller les logs du serveur Apache avec OPNsens à l'aide de Wazuh. Cela nous permet de détecter en temps réel :

- des erreurs serveur (erreurs 4xx/5xx),
- des tentatives d'intrusion (ex : scans, injection SQL, etc.),
- des accès non autorisés,
- ou encore des anomalies dans les fichiers de logs.

Ainsi, Apache devient une source d'événements que Wazuh peut analyser pour générer des alertes de sécurité.

# - Configuration de l’agent wazuh sur le serveur Apache

Sur le serveur à superviser (celui qui héberge Apache), nous devons vérifier :

### Vérification du service

```
sudo systemctl status apache2
```

### Vérification de la génération des logs

Les fichiers journaux par défaut sont :

```
/var/log/apache2/access.log
/var/log/apache2/error.log
```

## Configuration de l'agent Wazuh

Sur le même serveur, nous devons nous assurer que l'agent Wazuh est bien installé et communique avec le manager :

```
sudo systemctl status wazuh-agent
```

Configurons le fichier **/var/ossec/etc/ossec.conf** pour préciser l'adresse de wazuh :

```
<server>
  <address>192.168.1.30</address> <!-- IP du Wazuh Manager -->
</server>
```

---

## Ajout du module Apache dans Wazuh

Wazuh fournit déjà un module de décodage spécifique pour Apache.

Il suffit d'ajouter une entrée dans la configuration de l'agent :

```
sudo nano /var/ossec/etc/ossec.conf
```

Ajoutons le bloc suivant à l'intérieur de **<ossec_config>** :

```
<localfile>
  <log_format>apache</log_format>
  <location>/var/log/apache2/access.log</location>
</localfile>

<localfile>
  <log_format>apache</log_format>
  <location>/var/log/apache2/error.log</location>
</localfile>
```

Puis redémarrer l'agent :

```
sudo systemctl restart wazuh-agent
```

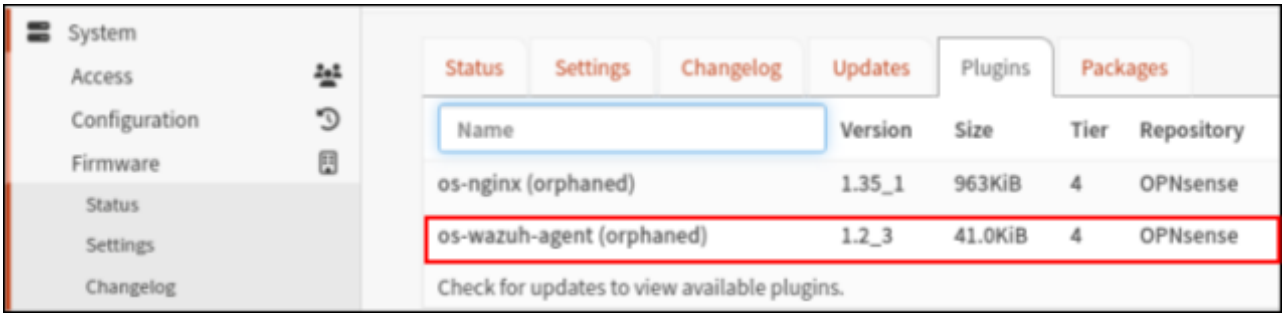
---

# - Installation et configuration de l'agent wazuh sur OPNsens :

**Méthode 1 : Via l'interface web OPNsense**

1. **Accéder au gestionnaire de paquets**
   - Menu : System → Firmware → Plugins
   - Rechercher : `os-wazuh-agent`
   - Cliquer sur le bouton **+** pour installer

Installation de l'agent-wazuh et nginx pour configurer le waf plus tard :

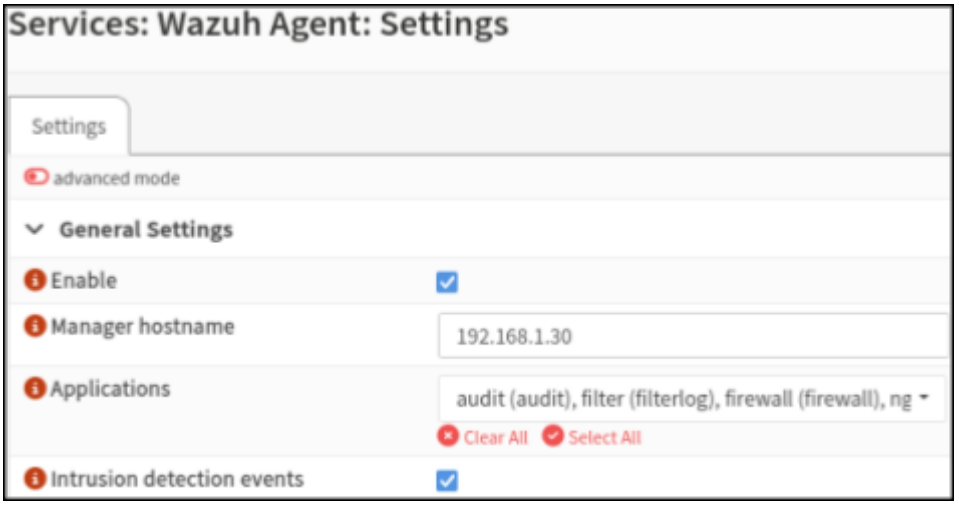


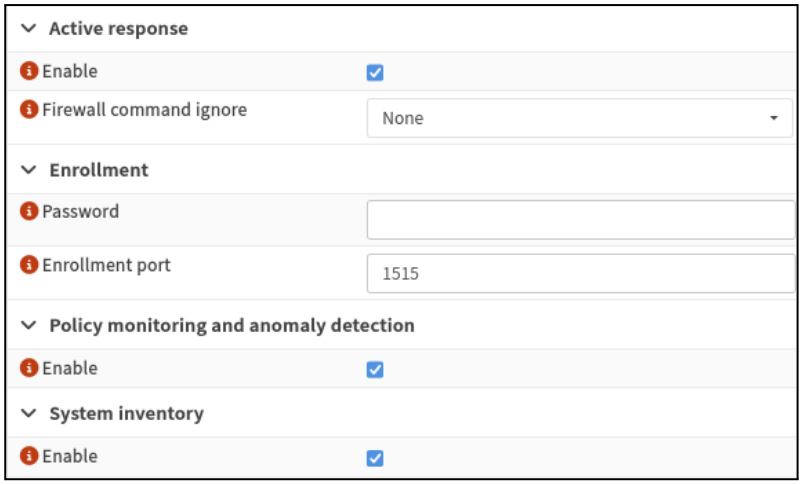

Configuration des settings de Wazuh agent :

## Configuration de l'agent Wazuh

**Via l'interface web OPNsense :**

1. **Accéder aux paramètres Wazuh**
   - Menu : Services → Wazuh Agent → Settings
2. **Configuration des paramètres principaux**
   - **Manager IP** : `192.168.1.30`
   - **Agent Name** : `OPNsense.internal` (ou nom personnalisé)
   - **Agent Groups** : `firewall,ids` (optionnel mais recommandé)
   - **Enable** : ✅ Cocher
   - Cliquer sur **Save**
3. **Démarrage du service**
   - Menu : Services → Wazuh Agent → Status
   - Cliquer sur **Start**




# Configuration de la collecte des logs

**Identification des fichiers de logs OPNsense :**

Ici nous allons récupérer le chemin des fichier de stockage des logs d'OPNsens et mettre ces chemins dans /var/ossec/etc/ossec.conf et les logs seront envoyer à wazuh :

Ici on va prendre 192.168.1.1.error.log et waf_denied.access.log pour les intrusions (injection sql, etc…) :

```
root@OPNsense:~ # ls /var/log/nginx/
192.168.1.1.access.log          nginx_20251001.log   nginx_20251017.log
192.168.1.1.error.log  nginx_20251002.log   nginx_20251028.log
access.log                      nginx_20251007.log       waf_denied.access.log
permanentban.access.log
```

Dans le fichier /var/ossec/etc/ossec.conf on peut rajouter :

```
<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/nginx/192.168.1.1.error.log</location>
</localfile>

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

**Explication des formats de logs :**

- `syslog` : Format standard Unix (date, host, message)

Puis redémarrer l'agent wazuh en ligne de commande sur :

```
root@OPNsense:~ # service wazuh-agent restart
```

---

## - Configuration de l'agent wazuh sur L'Active Directory :

# Configuration de l'agent Wazuh sur Active Directory — Version améliorée

### Contexte et objectifs

La supervision d'**Active Directory (AD)** est essentielle pour la sécurité d'une infrastructure.
AD :

- Centralise l'authentification et la gestion des utilisateurs.
- Constitue une cible majeure pour les attaquants.
- Contient des données sensibles (comptes, groupes, stratégies).

### Événements à surveiller

- Tentatives de connexion échouées (**Event ID 4625**)

- Création ou suppression de comptes (**Event ID 4720**, **4726**)
- Modification de groupes à privilèges (**Event ID 4732**, **4733**)
- Changements dans les politiques de sécurité
- Activités suspectes (ex. **Pass-the-Hash**, **Golden Ticket**)

Installation en une commande

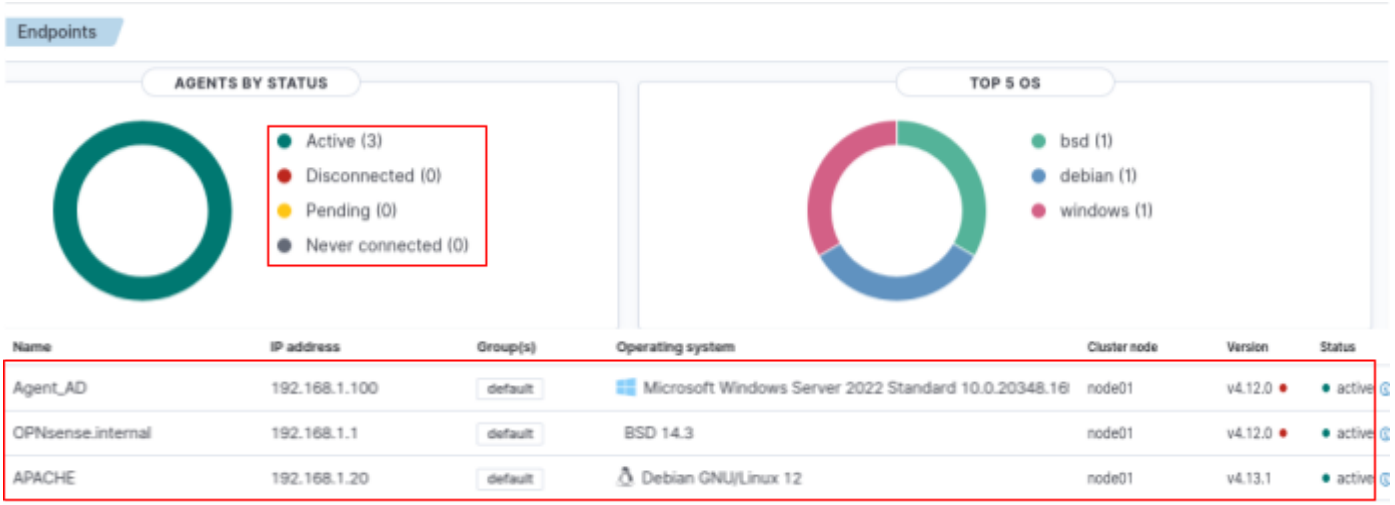
```
C:\Users\System32>Invoke-WebRequest -Uri
https://packages.wazuh.com/4.x/windows/wazuh-agent-4.12.0-1.msi -OutFile
$env:tmp\wazuh-agent.msi
msiexec.exe /i $env:tmp\wazuh-agent.msi /q WAZUH_MANAGER="192.168.1.30"
WAZUH_AGENT_NAME="Agent_AD"
C:\Users\gen97>net start wazuh
```

**Explication des paramètres :**

- `/i` : Mode installation
- `/q` : Mode silencieux (pas d'interface graphique)
- `WAZUH_MANAGER` : Adresse IP du manager Wazuh
- `WAZUH_AGENT_NAME` : Nom unique de l'agent (visible dans le dashboard)
- `WAZUH_REGISTRATION_SERVER` : Serveur pour l'enregistrement initial

---

Liste des nouveaux agents ajouté à wazuh :




---

# Configuration des metrics pour le dashboard OpnSense avec Wazuh

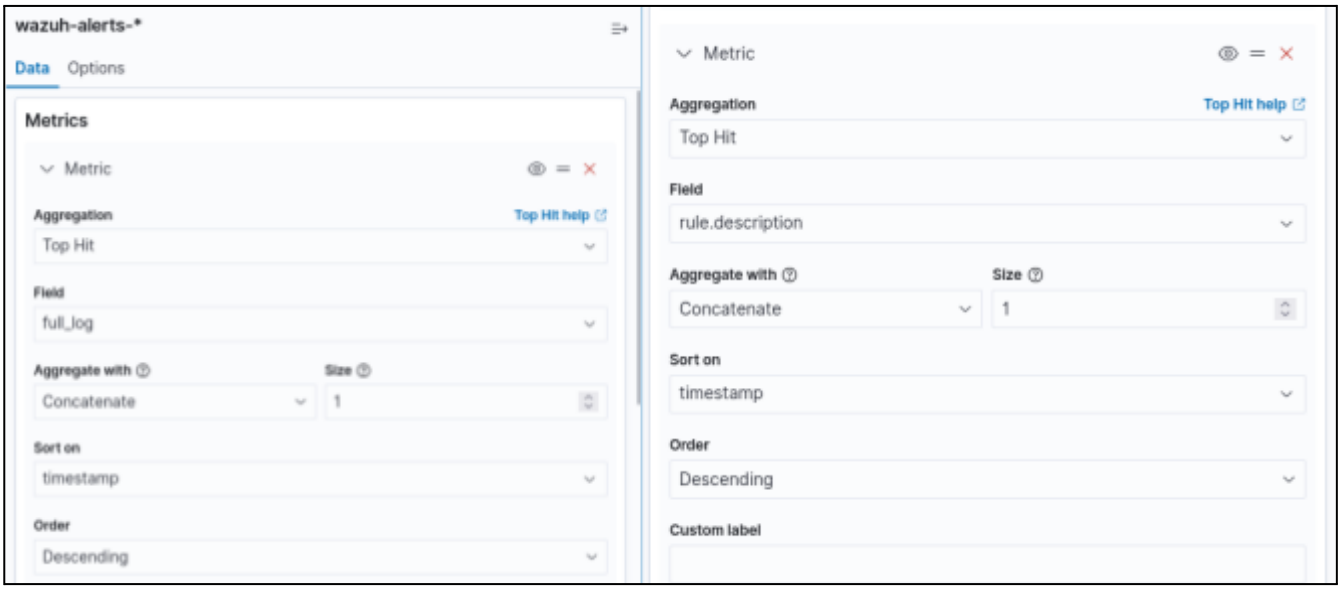
Pour afficher les alertes Wazuh sur le dashboard OpnSense, nous avons configuré deux **metrics** principales :

1. **Metric : full_log**
    - Récupère le contenu complet du dernier log (**`full_log`**).
    - Trié par date (**`timestamp`**) du plus récent au plus ancien.
    - Permet d'afficher directement le dernier log généré par Wazuh.
2. **Metric : rule.description**
    - Récupère la description de la règle Wazuh qui a déclenché l'alerte (**`rule.description`**).
    - **Trié par date (timestamp)** du plus récent au plus ancien.
    - Affiche la dernière description d'alerte pour comprendre rapidement la nature de l'incident.

**Méthode utilisée :**

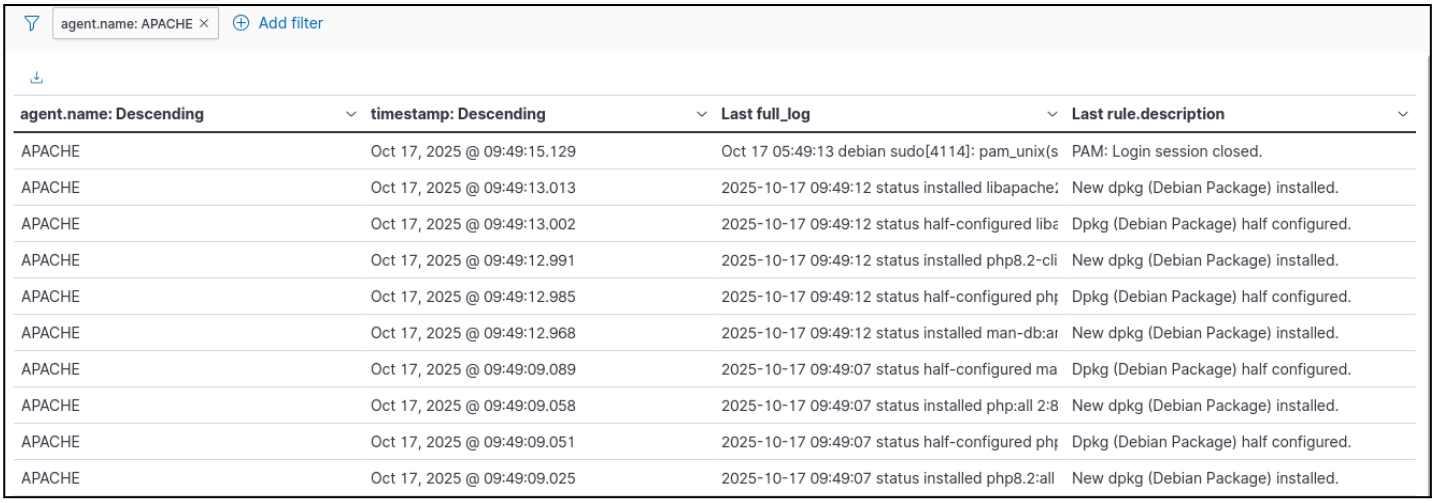
- Agrégation **Top Hit** : on ne récupère que la dernière valeur pertinente.
- Taille = 1, pour ne prendre que le dernier log ou description.

Cette configuration permet au dashboard d'afficher de manière claire et concise les alertes les plus récentes et leur description, facilitant le suivi et l'analyse des incidents.



# 4. Vérification de la remontée des logs et des logs d'attaque

- Dashboard Wazuh (section → **Security Events** → **Apache**).



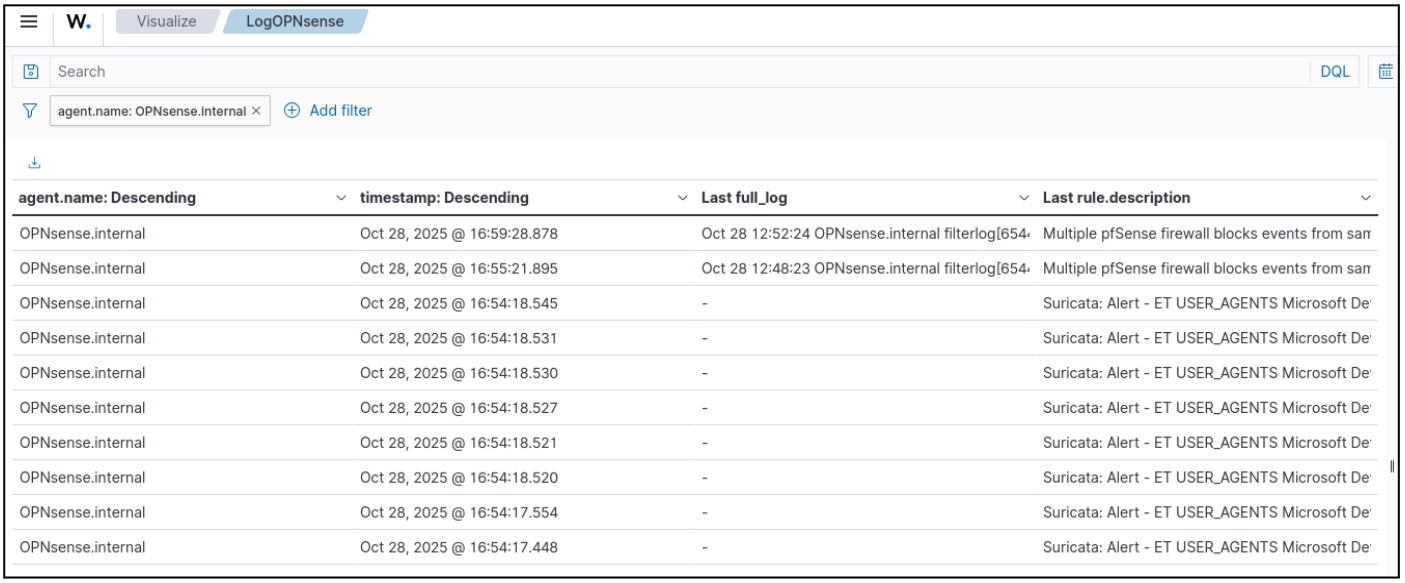

| agent.name: Descending | timestamp: Descending | Last full_log | Last rule.description |
|---|---|---|---|
| APACHE | Oct 17, 2025 @ 09:49:15.129 | Oct 17 05:49:13 debian sudo[4114]: pam_unix(s | PAM: Login session closed. |
| APACHE | Oct 17, 2025 @ 09:49:13.013 | 2025-10-17 09:49:12 status installed libapache2 | New dpkg (Debian Package) installed. |
| APACHE | Oct 17, 2025 @ 09:49:13.002 | 2025-10-17 09:49:12 status half-configured liba | Dpkg (Debian Package) half configured. |
| APACHE | Oct 17, 2025 @ 09:49:12.991 | 2025-10-17 09:49:12 status installed php8.2-cli | New dpkg (Debian Package) installed. |
| APACHE | Oct 17, 2025 @ 09:49:12.985 | 2025-10-17 09:49:12 status half-configured php | Dpkg (Debian Package) half configured. |
| APACHE | Oct 17, 2025 @ 09:49:12.968 | 2025-10-17 09:49:12 status installed man-db:ar | New dpkg (Debian Package) installed. |
| APACHE | Oct 17, 2025 @ 09:49:09.089 | 2025-10-17 09:49:07 status half-configured ma | Dpkg (Debian Package) half configured. |
| APACHE | Oct 17, 2025 @ 09:49:09.058 | 2025-10-17 09:49:07 status installed php:all 2:8 | New dpkg (Debian Package) installed. |
| APACHE | Oct 17, 2025 @ 09:49:09.051 | 2025-10-17 09:49:07 status half-configured php | Dpkg (Debian Package) half configured. |
| APACHE | Oct 17, 2025 @ 09:49:09.025 | 2025-10-17 09:49:07 status installed php8.2:all | New dpkg (Debian Package) installed. |

Résumé :
L'agent **APACHE** a procédé à l'installation ou la mise à jour de paquets **Apache** et **PHP** sur Debian le **17/10/2025 vers 09:49**.
Les logs suivants le confirment :

- `status installed libapache: New dpkg (Debian Package) installed.`
- `status installed php8.2-cli New dpkg (Debian Package) installed.`
- `status installed php8.2:all New dpkg (Debian Package) installed.`
- `status installed php:all 2:8 New dpkg (Debian Package) installed.`
- `status installed man-db:ar New dpkg (Debian Package) installed.`

Le log **`sudo[4114]: pam_unix(...) Login session closed`** indique qu'une **session administrative** a été utilisée, probablement pour exécuter ces installations.
→ Conclusion : installation normale de composants Apache/PHP effectuée avec privilèges **sudo**, sans erreur apparente.

- Dashboard Wazuh (section → **Security Events** → **OPNsens**).




| agent.name: Descending | timestamp: Descending | Last full_log | Last rule.description |
|---|---|---|---|
| OPNsense.internal | Oct 28, 2025 @ 16:59:28.878 | Oct 28 12:52:24 OPNsense.internal filterlog[654 | Multiple pfSense firewall blocks events from san |
| OPNsense.internal | Oct 28, 2025 @ 16:55:21.895 | Oct 28 12:48:23 OPNsense.internal filterlog[654 | Multiple pfSense firewall blocks events from san |
| OPNsense.internal | Oct 28, 2025 @ 16:54:18.545 | - | Suricata: Alert - ET USER_AGENTS Microsoft De |
| OPNsense.internal | Oct 28, 2025 @ 16:54:18.531 | - | Suricata: Alert - ET USER_AGENTS Microsoft De |
| OPNsense.internal | Oct 28, 2025 @ 16:54:18.530 | - | Suricata: Alert - ET USER_AGENTS Microsoft De |
| OPNsense.internal | Oct 28, 2025 @ 16:54:18.527 | - | Suricata: Alert - ET USER_AGENTS Microsoft De |
| OPNsense.internal | Oct 28, 2025 @ 16:54:18.521 | - | Suricata: Alert - ET USER_AGENTS Microsoft De |
| OPNsense.internal | Oct 28, 2025 @ 16:54:18.520 | - | Suricata: Alert - ET USER_AGENTS Microsoft De |
| OPNsense.internal | Oct 28, 2025 @ 16:54:17.554 | - | Suricata: Alert - ET USER_AGENTS Microsoft De |
| OPNsense.internal | Oct 28, 2025 @ 16:54:17.448 | - | Suricata: Alert - ET USER_AGENTS Microsoft De |

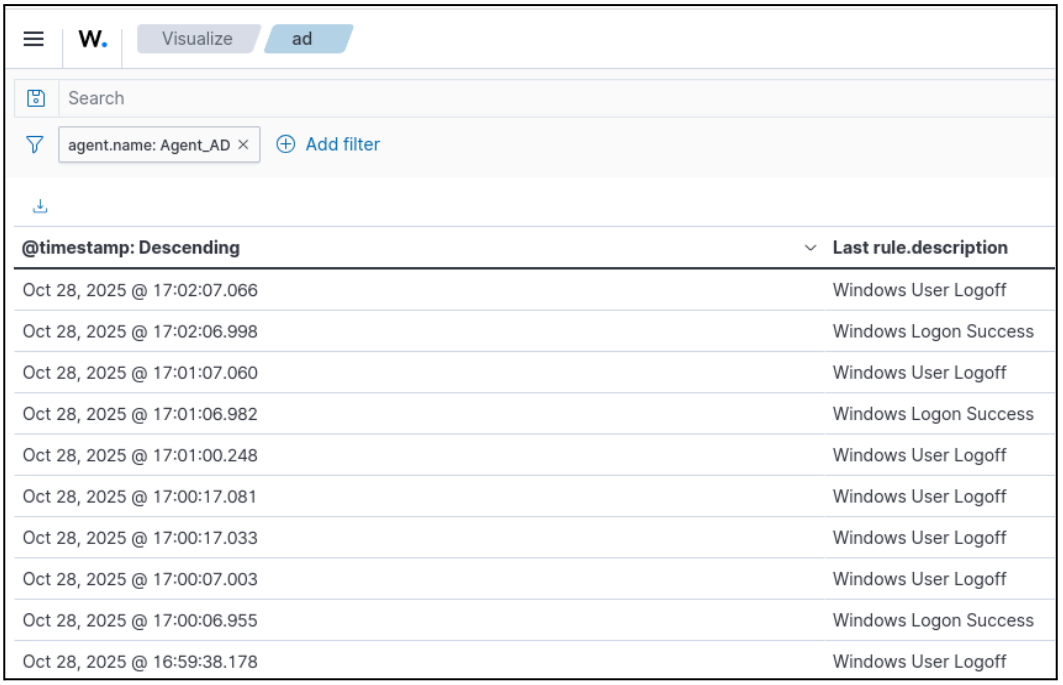
Résumé :
L'agent **OPNsense.internal** a généré des logs de **pare-feu** et de **Suricata** (dont la configuration est montrée juste après) le **28/10/2025** entre **16:54 et 16:59**.

Éléments clés :

- Entrées `filterlog` → plusieurs **blocages de trafic réseau** par le pare-feu (pfSense/OPNsense).
- Entrées `Suricata: Alert - ET_USER_AGENTS Microsoft…` → **alertes IDS** liées à des requêtes HTTP identifiées comme venant d'un **agent utilisateur Microsoft**, probablement du trafic sortant ou de tests de détection.

→ Conclusion : activité réseau surveillée et bloquée par le pare-feu, avec détections répétées de Suricata. Aucune anomalie critique, mais vigilance sur les connexions HTTP détectées.

- Dashboard Wazuh (section → Security Events → AD).




Résumé :
L'agent **Agent_AD** enregistre une succession d'événements **Windows Logon Success** et **Windows User Logoff** le **28/10/2025 entre 16:59 et 17:02**.

Interprétation :
Ces logs indiquent des **connexions et déconnexions utilisateur normales** sur le contrôleur Active Directory.
→ Aucune anomalie ou échec d'authentification détecté, activité utilisateur standard.

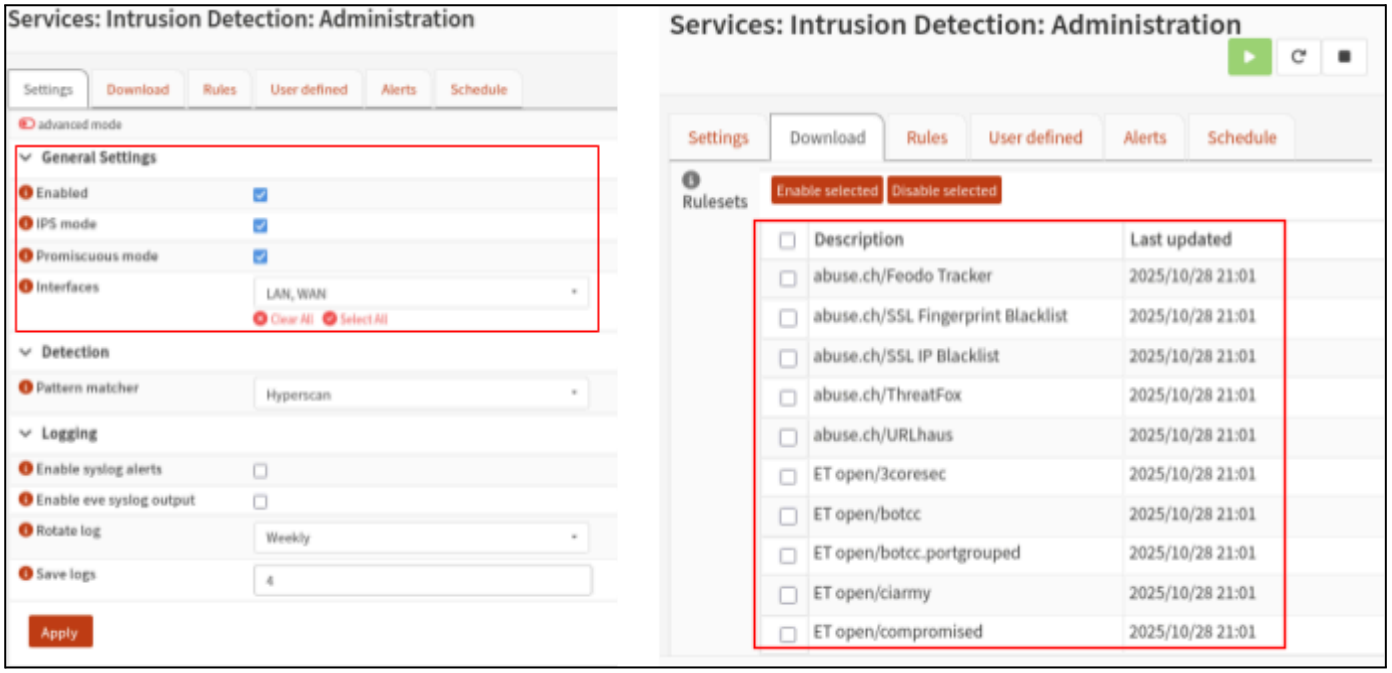
---

## Configuration IDS/IPS sur OPNsens :

Un IDS/IPS est un système de détection et prévention d'intrusions qui analyse le trafic réseau pour identifier et bloquer les comportements malveillants.

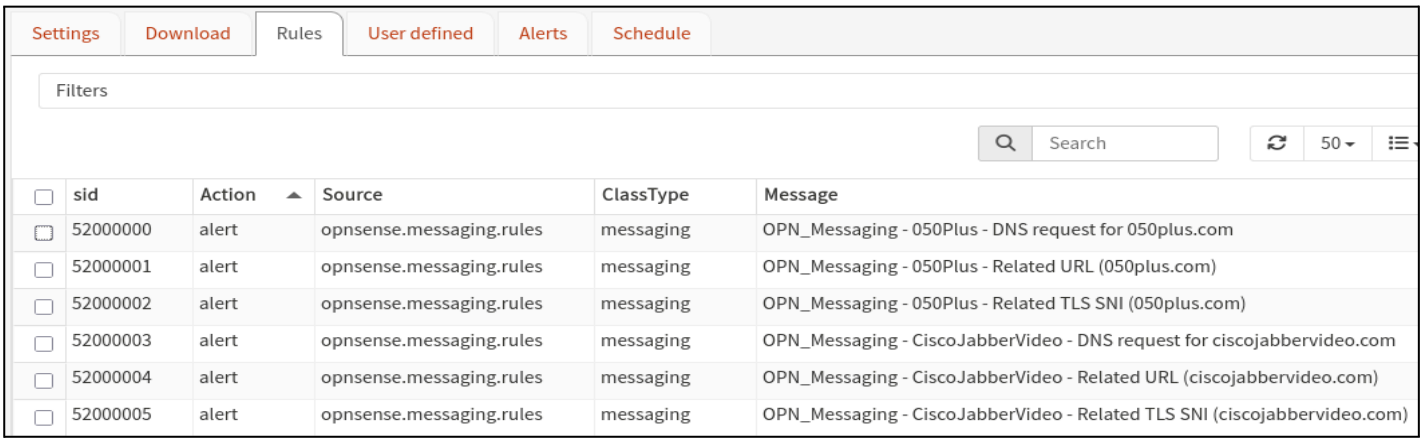
Suricata est un moteur IDS/IPS open-source capable d'analyser plusieurs Gbps de trafic. Il utilise des "règles" (signatures) pour identifier les menaces.

Configuration des settings du service IDS et activation du mode ips et installation des règles de détection d'intrusion :




Application des règles :

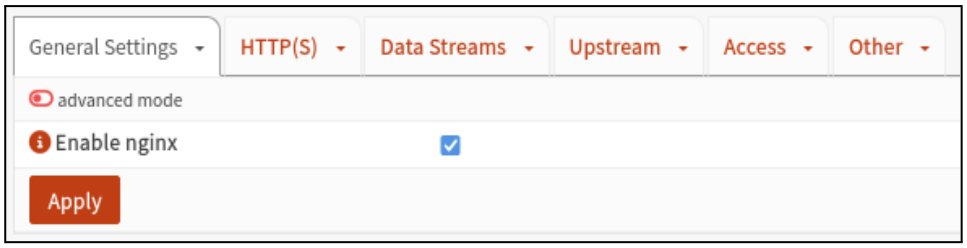
Ce screen montre que les **règles de détection sont bien appliquées sur OPNsense**. On y voit la liste des règles actives (type *User defined*) provenant du fichier `opnsense.messaging.rules`, avec leurs identifiants (SID), leurs actions (*alert*) et leurs descriptions. Cela confirme que la configuration de Suricata sur OPNsense est bien en place et que les règles sont correctement chargées et opérationnelles.



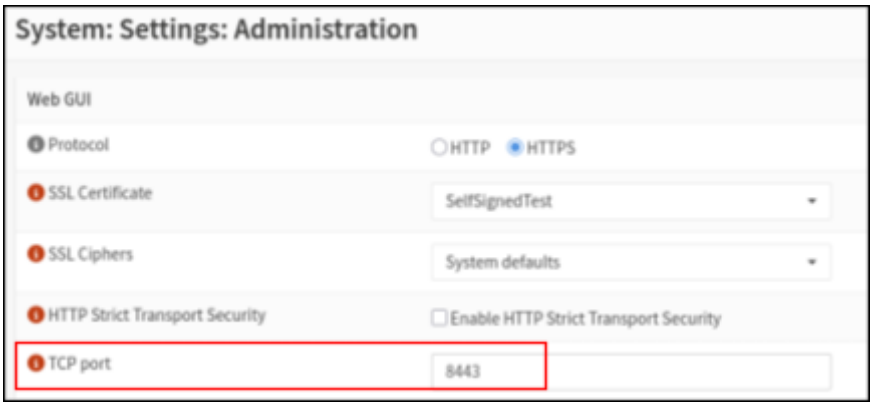
# Configuration WAF sur OPNsens :

Dans notre architecture, le WAF agit comme un bouclier protecteur devant notre serveur Apache, interceptant et analysant chaque requête avant qu'elle n'atteigne l'application web donc l'ip de OPNsens 192.168.1.1 redirigera automatiquement vers la page apache :

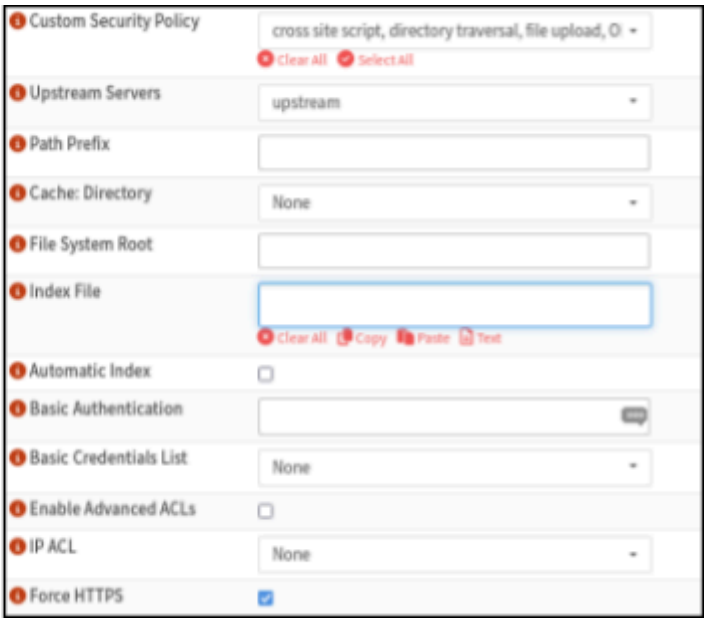
- En premier, activer nginx pour mettre en place le waf :




- Configuration du nouveau port 8443 pour accéder de nouveau au GUI du OPNsens :




La configuration d'une "Location" dans Nginx permet de définir comment le serveur doit traiter (protéger) les requêtes HTTP vers des URLs spécifiques :

Une **"Location" Nginx** définit comment le serveur gère les requêtes HTTP selon un **chemin d'URL**.
Elle permet de :

- Spécifier un **motif d'URL** et un **type de correspondance** (ex. `~*` pour insensible à la casse).
- Rediriger ou réécrire les URLs (**URL Rewriting**).
- Appliquer une **politique de sécurité personnalisée** (anti-XSS, anti-SQLi, contrôle d'accès IP, etc.).
- Définir un **upstream** (serveur cible), un **fichier d'index**, et activer ou non le **cache**.
- Forcer le **HTTPS** ou exiger une **authentification basique**.

→ En résumé : la section *Location* sert à contrôler la **protection et le routage** des requêtes vers des chemins précis du site.

---

Dans Nginx, un "HTTP Server" (aussi appelé "Virtual Host" ou "vhost") représente un site web complet avec ses paramètres d'écoute.



Un **HTTP Server** dans **Nginx** (ou *Virtual Host*) correspond à la configuration d'un **site web complet**.
Il définit :

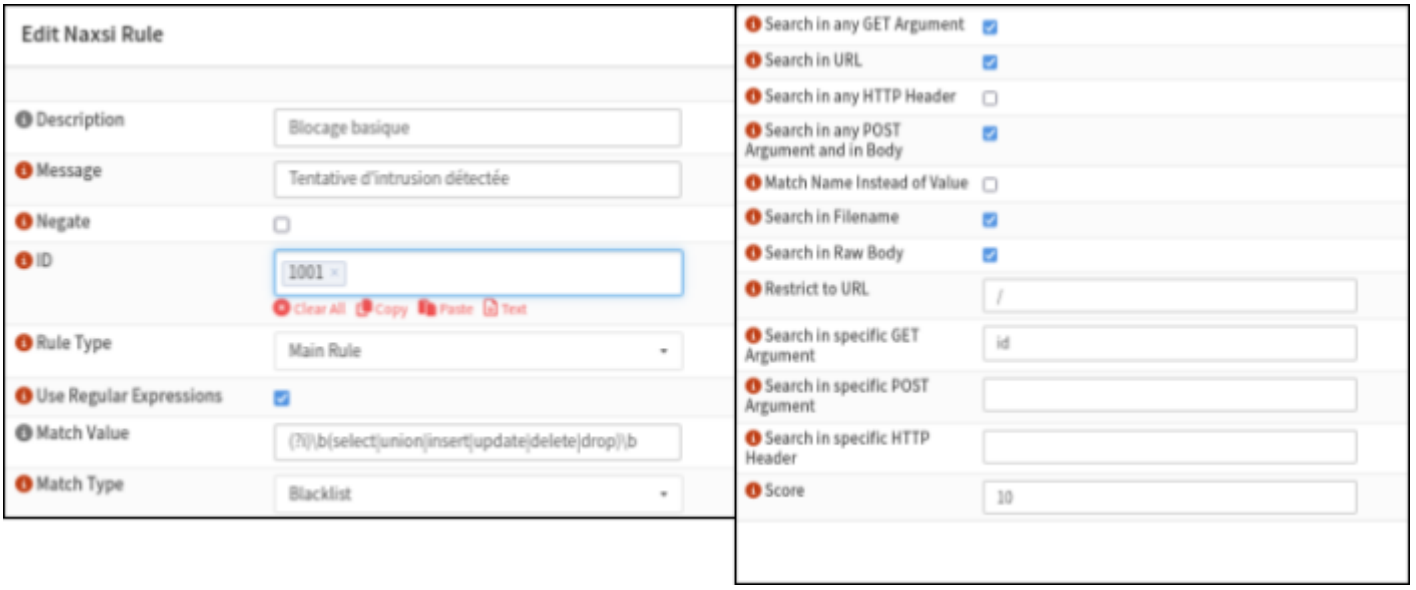
- **Adresse et port d'écoute** (`80` pour HTTP, `443` pour HTTPS).

- **Nom de domaine ou IP** du site (`server_name`).
- **Certificat TLS** pour le chiffrement HTTPS.
- **Répertoires racine** et **fichiers index**.
- **Règles de sécurité** (ACL IP, headers, limites de requêtes).
- **Authentification** (locale, backend externe).
- **Logs** et éventuellement **Let's Encrypt** pour le certificat automatique.
- Liste des **locations**, qui précisent comment traiter les chemins spécifiques.

→ En résumé : un *HTTP Server* gère **tout le comportement d'un site web dans Nginx** — écoute, sécurité, routage, et chiffrement.

---

NAXSI (Nginx Anti XSS & SQL Injection) fonctionne avec un système de règles qui définissent ce qui est considéré comme malveillant.




**NAXSI** est un module de sécurité pour **Nginx** qui agit comme un **pare-feu applicatif (WAF)**.
Il détecte et bloque les attaques web (ex. **XSS**, **injections SQL**) via un système de **règles**.

Chaque règle définit :

- Un **ID unique** (ex. `1001`)
- Un **type de recherche** (GET, POST, Header, URL, Body)
- Une **expression régulière** pour détecter les motifs dangereux

- Un **score de gravité** (ici 10)
- Une **action** (blocage, alerte, apprentissage)

Exemple : la règle `(?i)\b(select|union|insert|update|delete|drop)\b` bloque les tentatives d'injection SQL basées sur ces mots-clés.

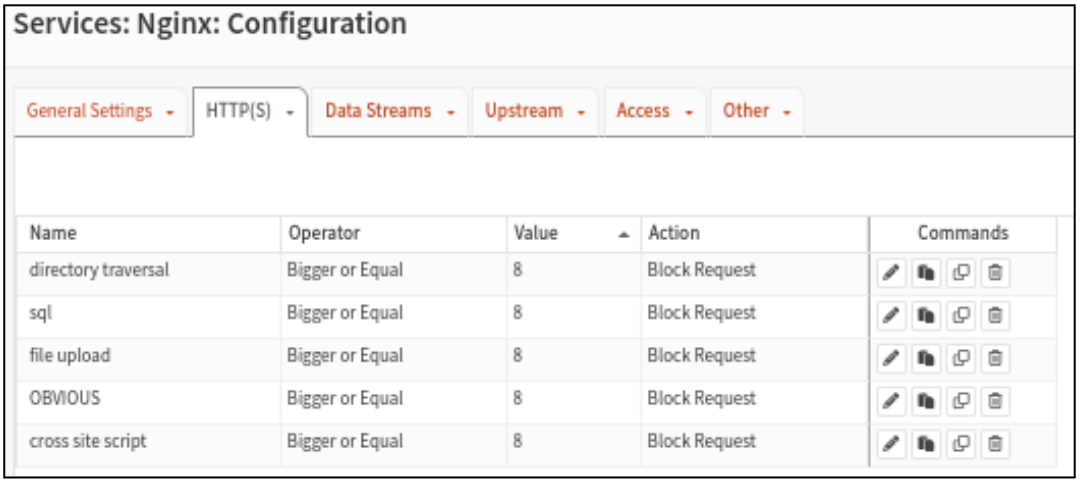
→ En résumé : NAXSI protège Nginx en analysant les requêtes et en bloquant celles correspondant à des **modèles d'attaques connus**.

---

**NAXsi WAF Policy :**

Une NAXSI WAF Policy est un PROFIL DE CONFIGURATION qui regroupe :

- Les règles de détection activées (MainRules)
- Les seuils de scoring pour bloquer les requêtes
- Les zones d'analyse (URL, ARGS, BODY, HEADERS)
- Le mode de fonctionnement (Learning, Blocking)
- Les WhitelistRules applicables




Une **NAXSI WAF Policy** définit la **stratégie de protection globale** appliquée par le pare-feu NAXSI sur Nginx.

Elle regroupe :

- Les **règles actives** (*MainRules*)
- Les **seuils de score** pour bloquer une requête (ex. `>= 8`)
- Les **zones analysées** : URL, paramètres, corps, en-têtes

- Le **mode d'opération** : *Learning* (apprentissage) ou *Blocking* (blocage)
- Les **whitelists** autorisant certains cas légitimes

Exemple de politique :

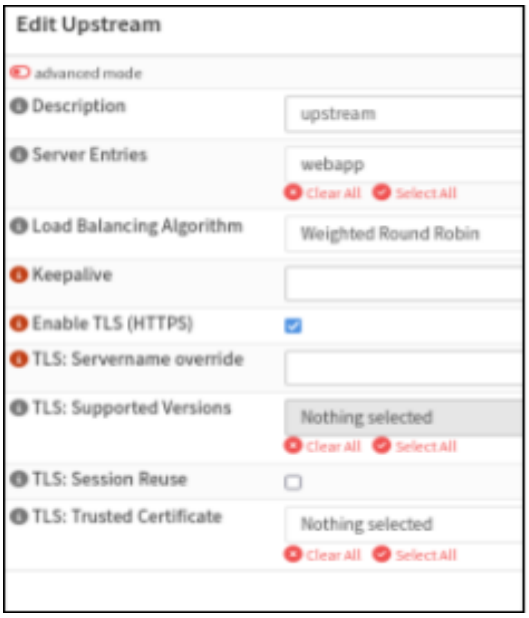
- `directory traversal ≥ 8 → Block Request`
- `sql ≥ 8 → Block Request`
- `file upload ≥ 8 → Block Request`
- `cross site script ≥ 8 → Block Request`

→ En résumé : la *WAF Policy* détermine **quand et comment NAXSI bloque une requête suspecte** selon le score cumulé de ses règles.

---

Ensuite nous allons configurer le "Upstream Server" est la définition d'un SERVEUR BACKEND réel qui héberge l'application web.

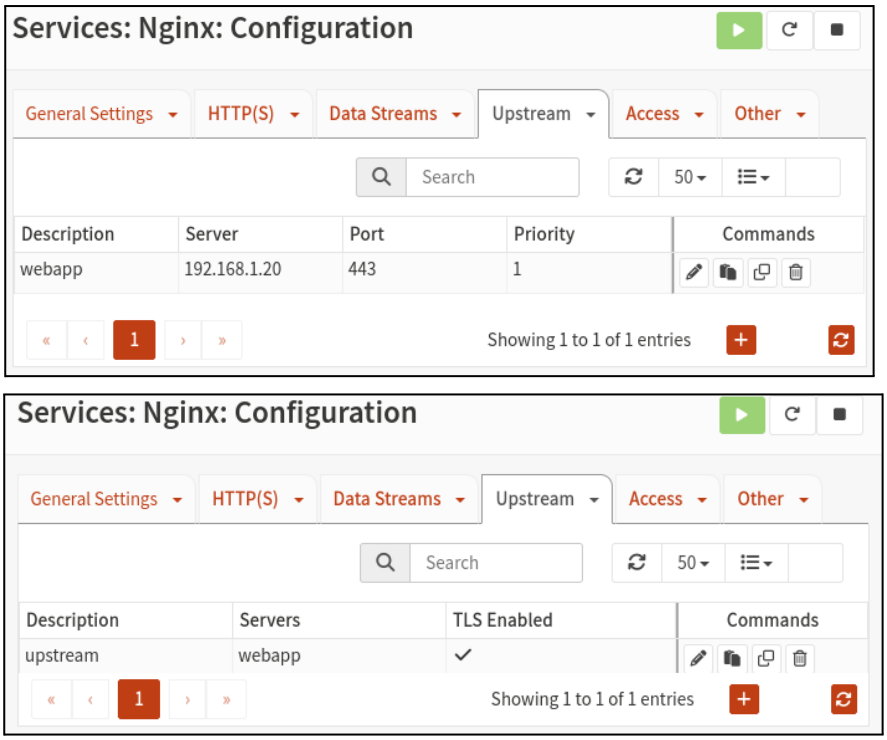
La configuration "Upstream Server" définit :





- OÙ se trouve le serveur backend (IP:Port)
- Comment s'y connecter (TCP, UNIX socket, etc.)
- Load balancing (si plusieurs serveurs)
- Timeouts et retry

Résultat après configuration de upstream :




Conclusion :
La configuration Nginx montre :

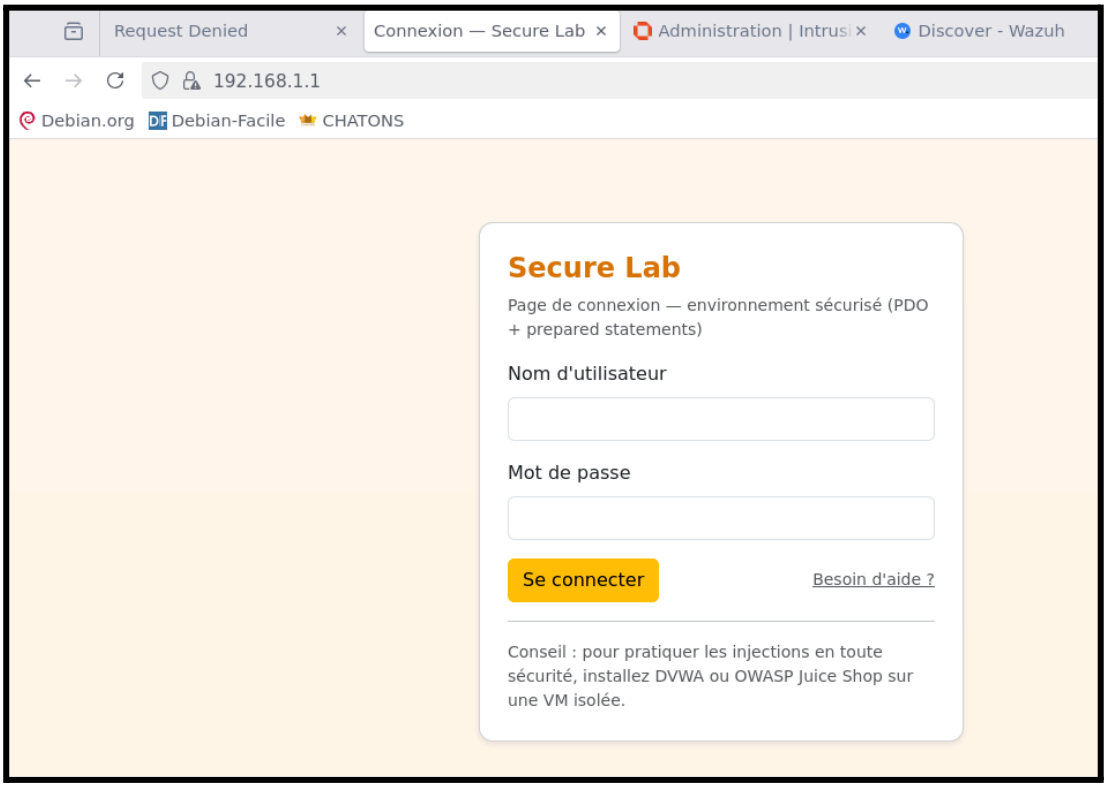
- Un **serveur HTTPS actif** (`192.168.1.20:443`) nommé **webapp**.
- Un **upstream** défini, également nommé **webapp**, servant de **backend applicatif**.
- Le **TLS est activé**, assurant une communication chiffrée.

→ En résumé : le service Nginx agit comme **reverse proxy sécurisé** vers une application backend, avec une configuration claire et fonctionnelle.

---

## Vérification du WAF :

Nous avons accédé à l'interface à l'adresse **192.168.1.1**. La page présentée — **Secure Lab / Intrusix — Discover - Wazuh** — est une page de connexion pour un environnement sécurisé crée pour le TP avec un **APACHE** et une **base de données**. Le formulaire comporte les champs **Nom d'utilisateur** et **Mot de passe** et un bouton **Se connecter**. La page indique l'utilisation de **PDO + prepared statements**, le WAF a donc été conçue pour limiter les injections SQL.
Pour des exercices d'attaque/sécurité en conditions contrôlées, il est conseillé d'utiliser des plateformes dédiées isolées (par ex. **DVWA** ou **OWASP Juice Shop** sur une VM).

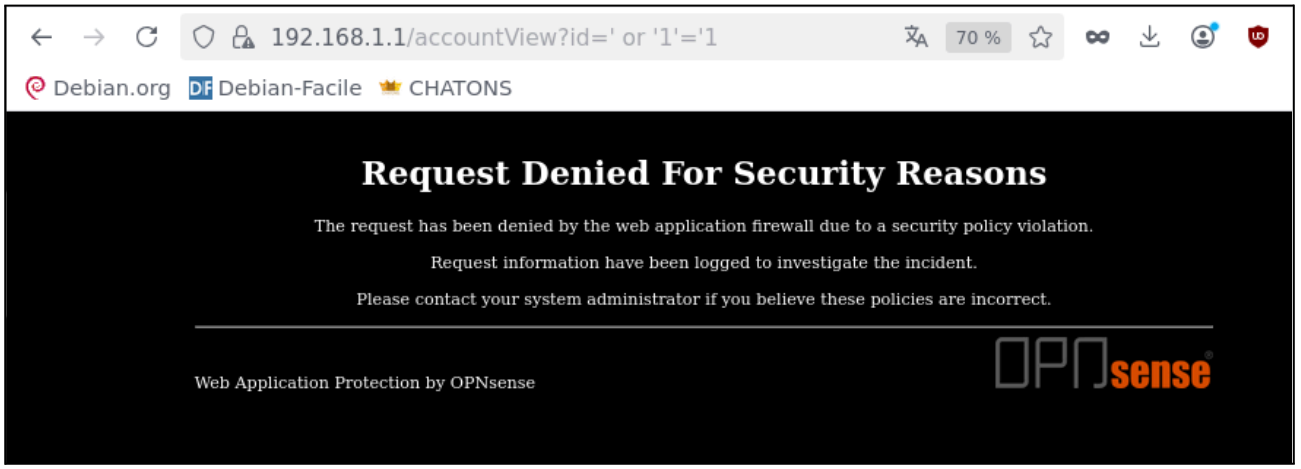



Test de plusieur type d'intrusion avec le NAXSI WAF de OPNsens :

- `http://www.mywebsite.com/accountView?id=' or '1'='1`

Nous avons envoyé la payload **id=' or '1'='1** — elle crée une **injection SQL** de type **tautologie** qui rend la condition toujours vraie et peut permettre d'afficher toutes les entrées ou de contourner une authentification sur une application vulnérable.

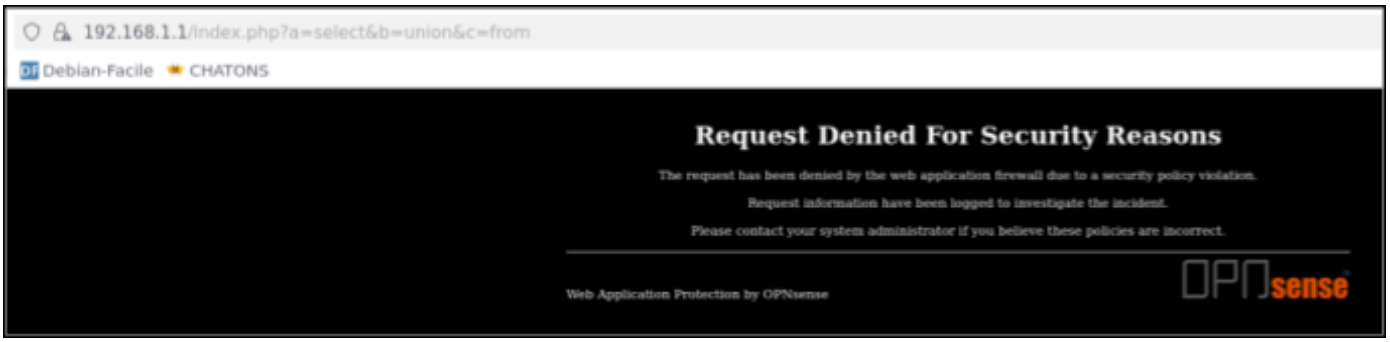
Nous testons cela pour vérifier l'efficacité du **WAF NAXSI** : et on peut constater que le waf bloque bien la requête malveillante avec **"Request Denied For Security Reasons" :**



- `http://192.168.1.1/index.php?a=select&b=union&c=from`

L'URL **http://192.168.1.1/index.php?a=select&b=union&c=from** simule une tentative d'injection SQL de type **UNION/SELECT** (les paramètres suggèrent l'utilisation de SELECT ... UNION SELECT ... FROM ...) afin d'essayer d'assembler et d'extraire des données de différentes tables.

Nous effectuons ce test pour vérifier si l'application ou le WAF (NAXSI/OPNsense) détecte et bloque ce type de requête et on peut constater que le waf bloque bien la requête malveillante avec **"Request Denied For Security Reasons" :**

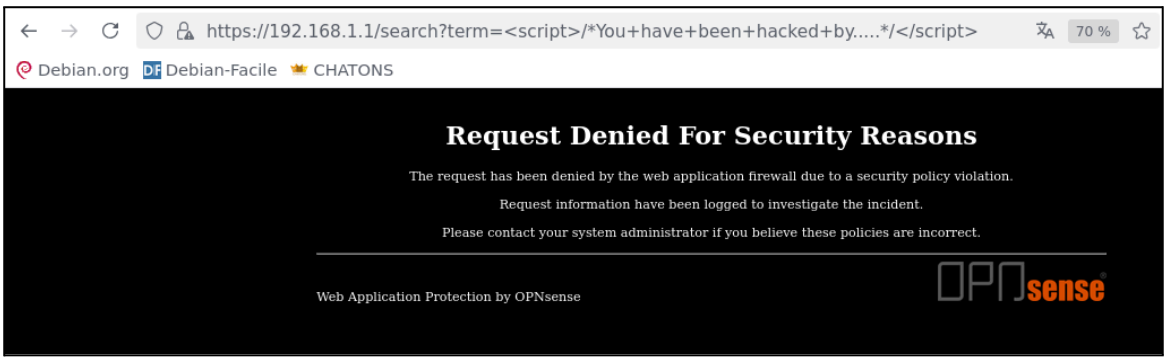



---

- `http://192.168.1.1/search?term=<script>/*You+have+been+hacked+by.....*/</script>`

Nous avons effectué un test avec l'URL http://192.168.1.1/search?term=<script>/*You+have+been+hacked+by.....*/</script>. Ce test simule une attaque XSS (Cross-Site Scripting) en injectant du code JavaScript dans un paramètre de recherche.

L'objectif est de vérifier si l'application ou le pare-feu applicatif (WAF NAXSI sous OPNsense) détecte et bloque ce type d'injection.et on peut constater que le waf bloque bien la requête malveillante avec **"Request Denied For Security Reasons" :**



# 6. Analyse des logs d'intrusion malveillante sur les Dashboards de wazuh

Depuis le Wazuh Dashboard (https://192.168.1.30), se connecter avec admin/admin.

Dans la section :

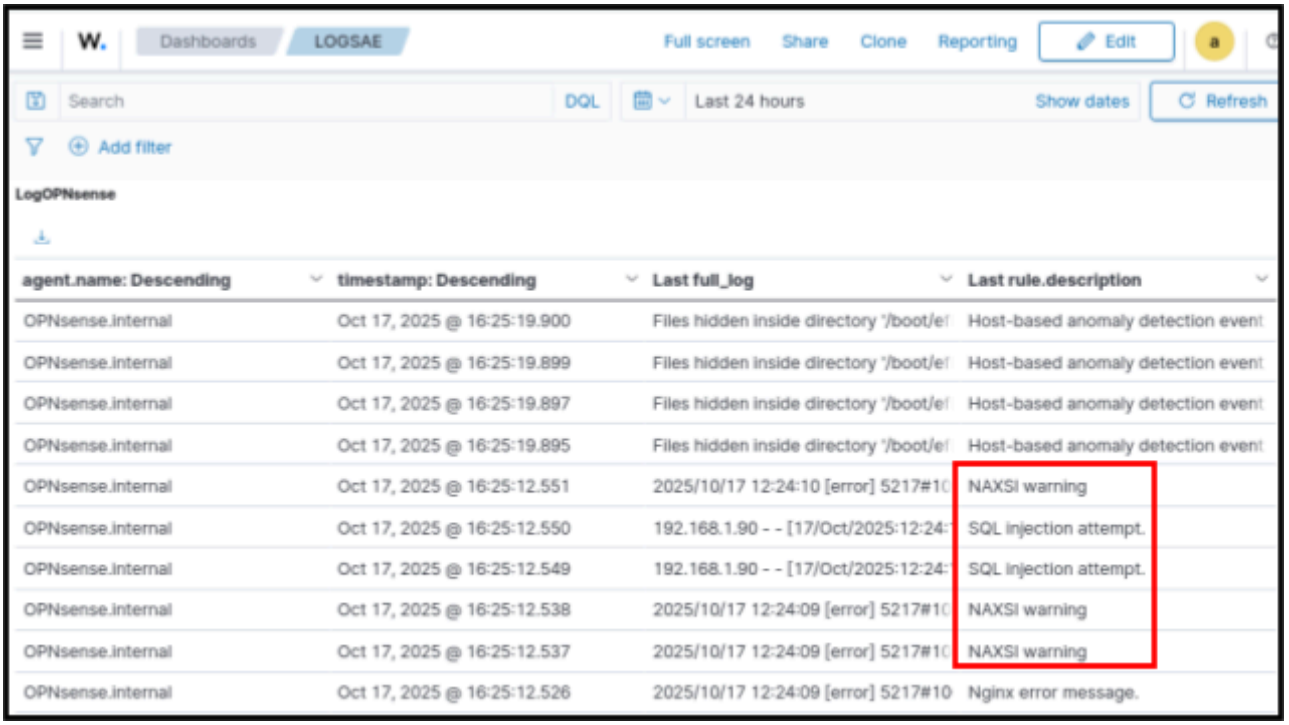
**Security events → OPNsens logs**

On visualise :

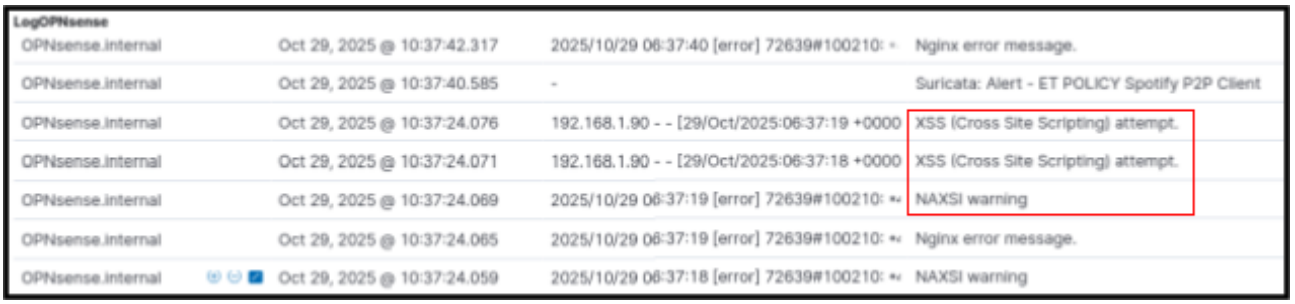
- Le nombre d'erreurs 404 ou 500
- Les adresses IP d'origine
- Les agents concernés
- Les types d'événements (malveillante) détectés

**Injection SQL vu dans wazuh:**

Ce **screen montre que les logs de l'attaque SQL sont bien remontés par Wazuh depuis OPNsense**. On y observe plusieurs entrées horodatées correspondant à des alertes générées lors de la tentative d'injection SQL. Les journaux indiquent notamment des messages du type *"SQL Injection attempt"* et *"NAXSI warning"*, confirmant que le pare-feu applicatif NAXSI a bien détecté et enregistré l'attaque. Cela prouve que la communication entre OPNsense et Wazuh fonctionne correctement et que la détection des intrusions est opérationnelle.

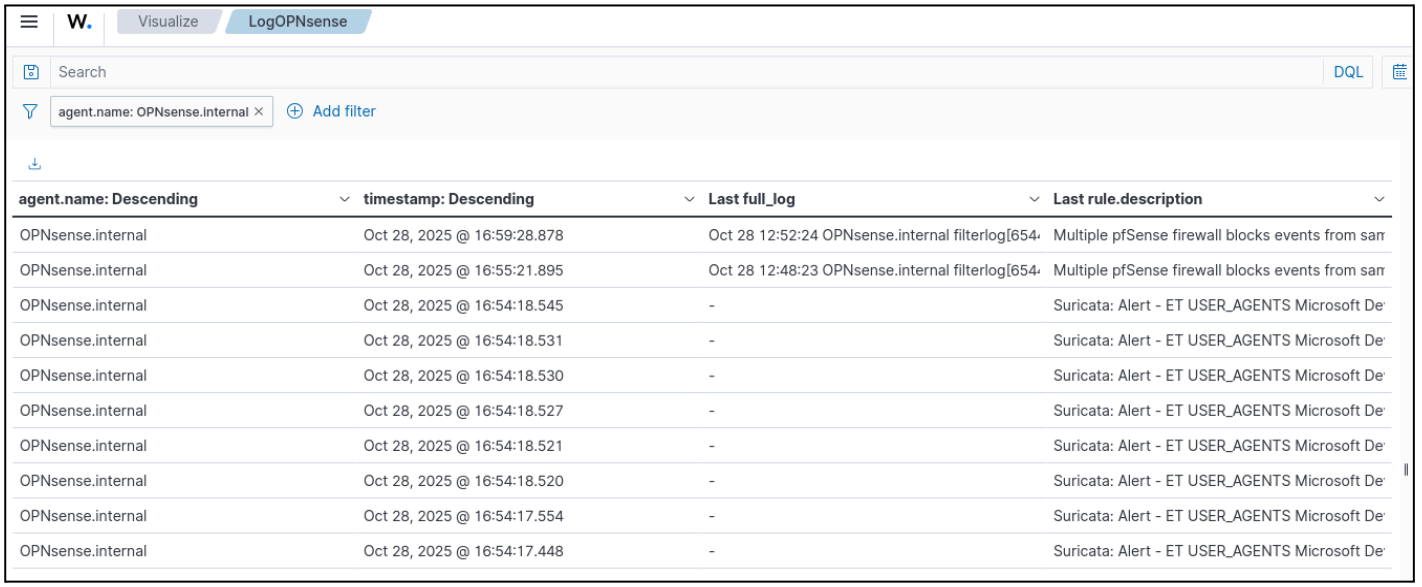


Ce journal montre que les logs des attaques XSS de la dernière attaque du WAF avec **“http://192.168.1.1/search?term=<script>/*You+have+been+hacked+by.....*/</script>”** qui sont bien détectés et remontés par Wazuh depuis OPNsense.




---

Ce **screen montre que les journaux de sécurité d’OPNsense sont bien collectés et affichés dans Wazuh**. Les logs indiquent plusieurs événements horodatés provenant de l’agent **OPNsense.internal,** notamment des alertes générées par **Suricata** (ex. *ET USER_AGENTS Microsoft*), ainsi que des **événements de blocage du pare-feu**.




Wazuh applique automatiquement ses règles de détection (OSSEC) pour classifier les alertes selon leur niveau de gravité.

# 7. Bénéfices

- Surveillance en temps réel du trafic Apache
- Détection automatique d'anomalies ou d'attaques web
- Centralisation dans un seul tableau de bord
- Historisation et corrélation avec d'autres sources (firewall, authentifications, etc.)

# Conclusion

Ce projet a permis de démontrer la mise en place d'une infrastructure complète de supervision et de défense en profondeur, répondant aux enjeux actuels de la cybersécurité.

**Sur le plan technique**, nous avons réussi à déployer une architecture Wazuh entièrement fonctionnelle avec chiffrement SSL/TLS de bout en bout, garantissant l'intégrité et la confidentialité des flux de données de sécurité. L'installation centralisée sur un serveur unique facilite la gestion tout en offrant une base évolutive vers une architecture distribuée si nécessaire.

**Sur le plan de la supervision**, l'intégration réussie de trois sources critiques (Apache, OPNsense et Active Directory) illustre la capacité de Wazuh à agréger et corréler des événements provenant d'environnements hétérogènes. Les tests d'intrusion réalisés (injections SQL, attaques XSS) ont confirmé l'efficacité du dispositif de détection, avec une remontée claire des alertes dans le tableau de bord Wazuh.

**Sur le plan de la protection active**, la mise en œuvre du WAF NAXSI et de l'IDS/IPS Suricata sur OPNsense a permis d'établir une défense multicouche devant le serveur Apache. Les règles de détection se sont avérées efficaces pour identifier et bloquer les tentatives d'exploitation de vulnérabilités web courantes.

**Sur le plan opérationnel**, l'intégration bonus avec Discord apporte une dimension collaborative moderne à la gestion des incidents. Cette automatisation permet une notification instantanée des équipes de sécurité, réduisant ainsi considérablement le temps de réponse face aux menaces.

# Partie Bonus – Intégration des alertes Wazuh avec Discord

Dans cette partie, j’ai réalisé une **intégration entre Wazuh et Discord** afin de recevoir automatiquement les alertes de sécurité dans un salon Discord. Cela permet de centraliser les notifications et de réagir rapidement en cas d’incident détecté par Wazuh.

## Création du Webhook Discord

1. J’ai ouvert **Discord** et sélectionné le serveur sur lequel je souhaitais recevoir les alertes.

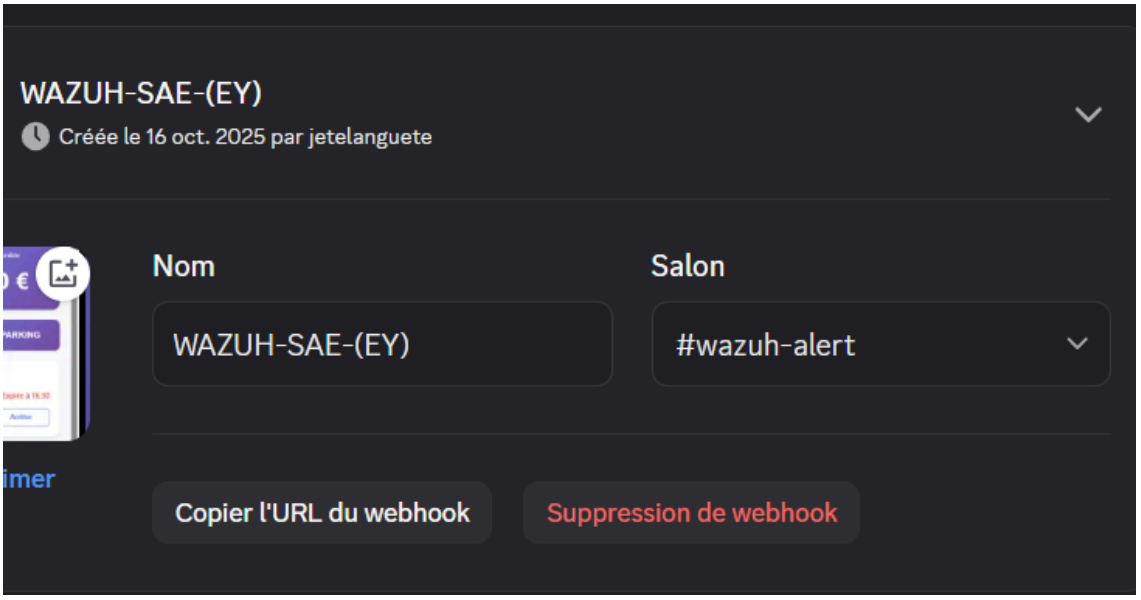
2. J’ai ensuite créé un **nouveau canal textuel**, nommé par exemple **#wazuh-alerts**.

3. Dans les **paramètres du serveur**, je suis allé dans la section **Intégrations** puis j’ai cliqué sur **Créer un Webhook**.

4. J’ai cliqué sur **Nouveau Webhook**, puis je lui ai donné le nom **WazuhAlerts** et j’ai sélectionné le canal **#wazuh-alerts.**

5. J’ai ensuite copié l’URL du Webhook pour l’utiliser plus tard dans la configuration de Wazuh.

**Webhook créer dans Discord:**




## Configuration du Dashboard Wazuh

1. Je me suis connecté à l'interface **Wazuh Dashboard**.
2. J'ai accédé à l'emplacement suivant :
   **Server Management** → **Settings**.
3. En haut à droite, j'ai cliqué sur **Edit configuration**.
4. Dans le fichier de configuration, j'ai ajouté le bloc suivant **entre les balises <global> et </global>**

**Fichier de configuration**

Manager configuration

Edit **ossec.conf** of **Manager**

```
<ossec_config>
  <global>
    <jsonout_output>yes</jsonout_output>
    <alerts_log>yes</alerts_log>
    <logall>no</logall>
    <logall_json>no</logall_json>
    <email_notification>no</email_notification>
    <smtp_server>smtp.example.wazuh.com</smtp_server>
    <email_from>wazuh@example.wazuh.com</email_from>
    <email_to>recipient@example.wazuh.com</email_to>
    <email_maxperhour>12</email_maxperhour>
    <email_log_source>alerts.log</email_log_source>
    <agents_disconnection_time>15m</agents_disconnection_time>
    <agents_disconnection_alert_time>0</agents_disconnection_alert_time>
    <update_check>yes</update_check>
  </global>
  <!--
  <integration>
    <name>custom-discord</name>
    <hook_url>https://discord.com/api/webhooks/1428332410751291402/3JXSUDQOiFDYoV54X5fq6uqD21GAG7ZGniozUYSD4bFaQF0BIoM0WRP-zt3x9o7l7EyG</hook_url>
    <alert_format>json</alert_format>
  </integration>
  -->

```

À noter que, sur le screen ci-dessus, la ligne de code apparaît en commentaire. C'est normal : pour qu'elle fonctionne, il suffit de la **retirer du commentaire**. Ce comportement est tout à fait normal dans la configuration.J'ai laissé la ligne de code en commentaire afin de pouvoir l'activer facilement lors de futurs tests d'alerte. Il me suffira simplement de retirer le commentaire pour que la règle soit prise en compte.

Accès au répertoire des intégrations :

```
cd /var/ossec/integrations
```

1.

Téléchargement des scripts d'intégration Discord :

```
wget https://raw.githubusercontent.com/maikroservice/wazuh-integrations/main/discord/custom-discord
wget https://raw.githubusercontent.com/maikroservice/wazuh-integrations/main/discord/custom-discord.py
```

2.

Vérification de la présence des fichiers :

```
ls -l
```

3.

Attribution des permissions nécessaires :

```
sudo chmod 750 /var/ossec/integrations/custom-*
sudo chown root:wazuh /var/ossec/integrations/custom-*
```

4. (Les fichiers apparaissent maintenant en vert, indiquant qu'ils sont exécutables.)

## Installation des dépendances Python

Pour permettre l'exécution du script Python d'intégration, j'ai installé pip3 et la bibliothèque requests :

```
sudo apt-get install python3-pip
pip3 install requests
```

## Redémarrage de Wazuh

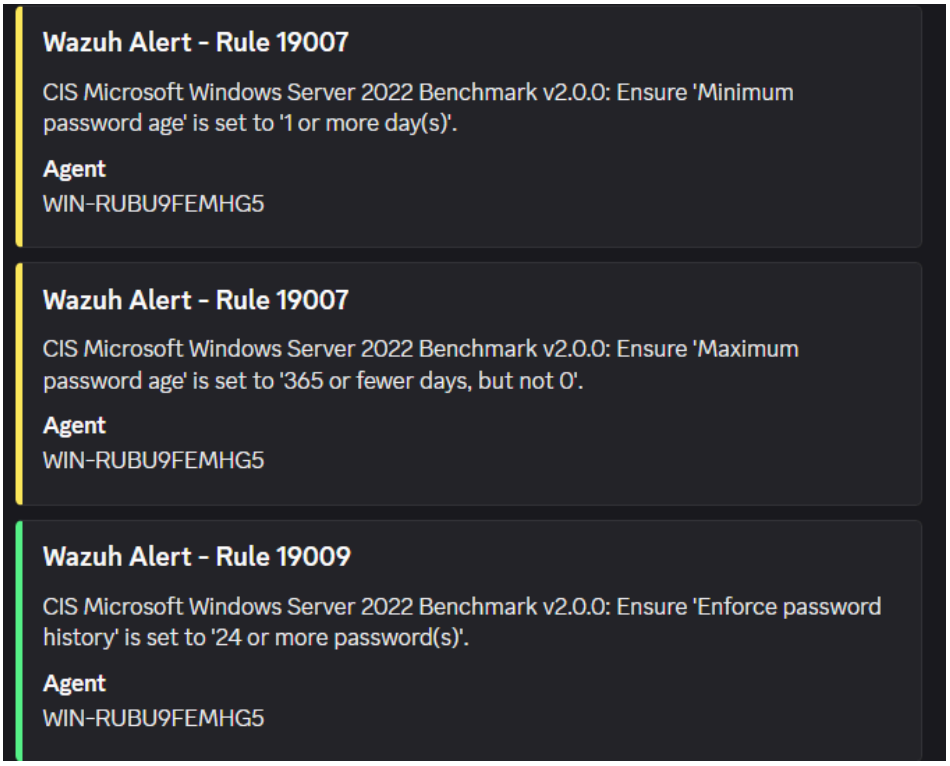
Enfin, j'ai redémarré le service Wazuh pour appliquer la configuration :

```
/var/ossec/bin/wazuh-control restart
```

## Vérification

Une fois le service redémarré, un **message de confirmation** s'est affiché directement sur le canal Discord configuré, prouvant que l'intégration fonctionne correctement.
Désormais, toutes les alertes de sécurité détectées par Wazuh seront automatiquement envoyées dans Discord en temps réel.

**Exemple d'alerte afficher sur Discord:**




# Conclusion – Intégration webhook Discord

La mise en place de l'intégration entre **Wazuh et Discord** s'est révélée être une solution simple, efficace et moderne pour la **centralisation des alertes de sécurité**.
Grâce à la création d'un **webhook Discord** et à la configuration du module **custom-discord** dans Wazuh, il est désormais possible de recevoir **en temps réel** toutes les notifications critiques directement dans un salon Discord dédié.

Cette intégration améliore considérablement la **réactivité face aux incidents** et permet un **suivi collaboratif** entre les membres de l'équipe, sans avoir à se connecter en permanence au tableau de bord Wazuh.
Elle démontre également la **souplesse de Wazuh**, capable de s'adapter à différents outils de communication et d'automatisation.

En résumé, cette partie bonus prouve qu'il est possible d'allier **sécurité, automatisation et praticité** grâce à une intégration légère mais très utile dans un environnement de supervision moderne.