# SOMMAIRE



# Introduction

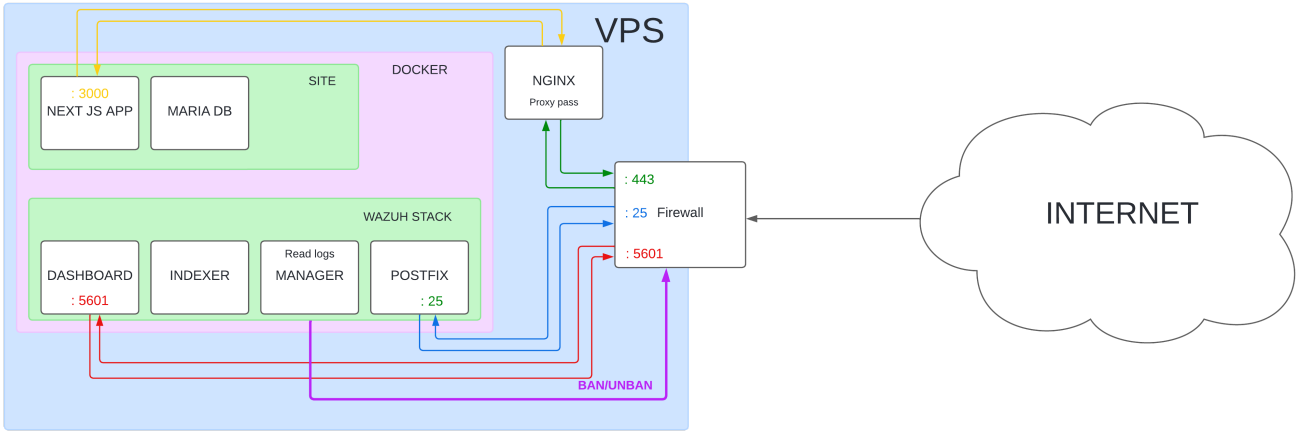
Voici le projet fil rouge du labo SSI de l'année scolaire 2023/2024 réalisé par Axel BROQUAIRE et Hugo ANDRIAMAMPIANINA. Il s'agit de coder un site internet classique où les utilisateurs peuvent se connecter/s'inscrire, puis laisser un message aux administrateurs grâce à un formulaire de contact. Les administrateurs peuvent ensuite se connecter à leurs propres comptes pour visionner les messages et l'image qui y est potentiellement jointe.

Au-delà de la simple fonctionnalité du site, une attention particulière a été portée à la sécurité de l'infrastructure qui héberge le site. Un système de monitoring de sécurité (Wazuh) avec des alertes par mail pour la gestion des événements et Suricata pour la détection d'intrusions réseau, ont été mis en place pour garantir la surveillance constante de l'environnement. De plus, des mesures de durcissement de configuration et de l'host ont été implémentées afin de renforcer la résilience face aux menaces potentielles.

## Specs

- OS : Ubuntu 22.04
- 4Go RAM
- 4 cores CPU

# Schema de l'infrastructure



Ce schéma représente l'infrastructure de notre projet, toutes les connexions réseau passent initialement par un pare-feu centralisé qui agit comme une première ligne de défense en filtrant le trafic entrant et sortant.

Les utilisateurs normaux accèdent au site internet à travers le port 443 (HTTPS), assurant ainsi une communication chiffrée. Le trafic utilisateur est d'abord dirigé vers le pare-feu avant d'être transmis à NGINX. NGINX agit comme un reverse proxy, redirigeant ensuite les requêtes vers le conteneur Docker hébergeant le site web. Ce conteneur communique avec sa base de données au besoin et renvoie les réponses à NGINX, qui agit en tant qu'intermédiaire entre le programme et l'utilisateur final.

Les administrateurs accèdent au dashboard de Wazuh en se connectant directement au port 5601. Wazuh collecte des logs à partir de diverses sources, les analyse pour détecter d'éventuelles violations de sécurité et génère des alertes qui sont visualisées sur le dashboard de Wazuh. En cas d'alerte atteignant un niveau défini dans la configuration (dans notre cas, niveau 12), Wazuh déclenche l'envoi d'un e-mail d'alerte via Postfix. Wazuh supporte nativement l'envoie d'e-mail mais ne supporte aucun serveur smtp, j'ai donc ajouté postfix dans le docker compose de Wazuh pour permettre un envoie simple des e-mails tout en conteneurisant le serveur smtp.

Wazuh dispose également d'une fonctionnalité d'active-response, permettant la mise en œuvre de mesures correctives automatisées. Par exemple, en cas de détection d'une attaque de brute-force, Wazuh peut déclencher un bannissement temporaire pour l'adresse IP source.

# Description, installation, mise en place et documentation fonctionnelle

## Installation de Wazuh server, conteneurisé en single-node

Single-node : un seul server qui recevra les logs pour les traiter

```
git clone https://github.com/wazuh/wazuh-docker.git -b v4.7.3
sudo sysctl -w vm.max_map_count=262144
cd wazuh-docker/single-node/
```

Les identifiants par défaut :

```
Username : admin

Password : SecretPassword
```

Dans le cadre d'un déploiement sur un serveur connecté à Internet, il faut évidemment modifier tous les mots de passe par défaut, c'est donc ce que nous allons faire.

Pour générer les hashs des mots de passe, Wazuh nous fourni un conteneur. Il suffit de lancer la commande ci-dessous, il sera demandé un mot de passe à entrer puis on recevra son hash

```
docker run --rm -ti wazuh/wazuh-indexer:4.7.3 bash /usr/share/wazuh-
indexer/plugins/opensearch-security/tools/hash.sh
```

Les users à modifier se trouvent dans `config/wazuh_indexer/internal_users.yml`, on peut garder uniquement les users `admin` et `kibanaserver` car les autres sont des users de démo. Il faut donc mettre le(s) hash obtenu grâce a la commande précédente entre des guillements.

Il faut ensuite modifier le mot de passe de l'utilisateur de l'API qu'il faut mettre en clair entre des guillements dans `config/wazuh_dashboard/wazuh.yml`

Enfin, il faut modifier le `docker-compose.yml` et y mettre les mots de passe en clair (et les noms d'utilisateur si vous avez changé) aux lignes 24, 30, 81, 84 et 86.

Il faut ensuite générer les certificats et lancer la stack:

```
docker compose -f generate-indexer-certs.yml run --rm generator
docker compose up -d
```

Il ne reste plus qu'à aller aller sur l'IP de votre site pour accéder au dashboard !

⚠ N°1 : Il ne faut pas oublier de configurer le pare-feu en conséquence

⚠ N°2 : Par défaut le dashboard tourne sur le port 443, si utilisez ou comptez utiliser ce port pour un autre service (NGINX par exemple pour un site internet), il faut penser à le modifier dans le `docker-compose.yml`

## Enroller un serveur comme agent

Il suffit de suivre la procédure classique, c'est à dire se rendre sur le dashboard, aller dans `Agents` puis cliquer sur `Deploy new agent`, enfin remplir les champs demandés.

⚠ `Server address` réfère à l'IP de Wazuh server, donc la même IP que votre agent si vous déployez l'agent sur le même serveur que Wazuh serveur.

## Intégration de Suricata

### Installation

```
sudo add-apt-repository ppa:oisf/suricata-stable
sudo apt-get update
sudo apt-get install suricata -y
```

### Télécharger les rulesets de suricata Emerging Threats

```

```

```
cd /tmp/ && curl -LO https://rules.emergingthreats.net/open/suricata-
6.0.8/emerging.rules.tar.gz
sudo tar -xvzf emerging.rules.tar.gz && sudo mv rules/*.rules
/etc/suricata/rules/
sudo chmod 640 /etc/suricata/rules/*.rules
```

### Configuration de Suricata

Dans le fichier `/etc/suricata/suricata.yaml`, il faut modifier les paramètres suivants :

Mettre le `HOME_NET` avec l'IP complète. Par exemple :

```
HOME_NET: "135.125.238.153/32"
EXTERNAL_NET: "any"
```

Mettre la bonne interface à surveiller :

```
af-packet:
  - interface: ens3
```

Pour ce qui est des règles, il faut configurer le bon dossier de stockage des règles si ce n'est pas déjà fait et activer toutes les règles :

```
default-rule-path: /etc/suricata/rules

rule-files:
  - "*.rules"
```

### Intégration

- Créer un groupe nommé "Suricata" via le dashboard : `Management` => `Groups` => `Add new group`
- Agent l'agent souhaité dans ce groupe : `Management` => `Groups` => Cliquer sur le groupe => `Manage agents`
- Activer la surveillance du fichier de log de Suricata via le dashboard ou en modifier `/var/ossec/etc/shared/Suricata/agent.conf` depuis l'intérieur du conteneur de Wazuh manager :

```
<agent_config>
  <localfile>
    <log_format>json</log_format>
    <location>/var/log/suricata/eve.json</location>
  </localfile>
</agent_config>
```

Lancer Suricata

```
sudo systemctl start suricata
sudo systemctl enable suricata
sudo suricata-update
sudo systemctl restart suricata
```

⚠ Ne pas oublier de restart Wazuh

# Zoom sur le hardening et les configurations

- nginx
- ssh
- docker

# Problèmes rencontrés

- Docker rootles :

J'ai essayé de mettre en place docker "rootless" pour le démon, de sorte à ce que si un attaquant parvient à s'introduire dans le conteneur via la site internet par exemple puis à s'en extraire pour arriver sur la machine hôte, il n'obtienne pas les droits root.

J'ai réussi à le mettre en place pour les conteneurs du site mais pas pour Wazuh. J'ai par la suite découvert que Wazuh est incomptatible avec docker "rootless".

Il est possible d'avoir un démon à la fois en "rootless" et un autre qui fonctionne de manière classique, je voulais donc faire en sorte que Wazuh utilise la version classique et le site la version "rootless" mais je n'y suis pas parvenu.

- Système d'exploitation :

Initialement, le système d'exploitation du serveur qui avait été choisi était Debian 12. Mais Debian 12 (contrairement à Debian 11) n'utilise plus rsyslog, mais plutôt journalctl. Ce qui n'est pas compatible avec Wazuh, ou en tout cas pas sans configuration supplémentaire que je n'ai pas réussi à faire.

Nous sommes donc passé sur Debian 11. Mais les dépendances étant trop vieille et des problèmes avec Suricata (aucune règle ne fonctionnait, et une erreur me disant que la longueu des paquets est invalide tournait en boucle) m'ont convaincu de changer d'OS.

Nous somme enfin passé sur Ubuntu 22.04 et tout fonctionne maintenant.

# Axes d'amélioration

- Améliorer les règles de détection :

Pour accroître l'efficacité de la surveillance et de la détection des menaces, il serait intéressant de se pencher plus sur les règles de détection de Suricata, notamment en prenant en compte que le traffic qui est destiné au site et à Wazuh sont chiffrés.

- Mettre le dashboard de Wazuh derrière le reverse proxy de NGINX :

Il pourrait être intéressant de mettre le dashboard de Wazuh derrière le reverse proxy de NGINX pour bénéficier des différentes fonctionnalités de NGINX mais aussi facilité la gestion des sites accessibles depuis l'hôte.

- Docker rootles :

Pour empêcher qu'un assaillant, ayant réussi à pénétrer dans le conteneur Docker et à en échapper, puisse obtenir les privilèges root, il serait judicieux de basculer le démon Docker en mode "rootless". Cependant, étant donné que Wazuh ne prend pas en charge cette configuration, une solution envisageable serait de faire fonctionner un démon en mode "rootless" pour le conteneur du site, tandis qu'un démon classique serait maintenu pour Wazuh.

# Conclusion

# Annexes

Sources et fichiers de conf