

RAPPORT

SAÉ 5.CYBER.03 ASSURER LA SECURISATION ET LA SUPERVISION AVANCÉES D'UN SYSTEME

Présenté par

Hoareau Anthony

Phamsi Lillian

Bernard Didier

Pasqua Xavier

Encadré par

M.Fontaine Thibaut

Table des matières

1. Introduction	3
1.1. Contexte	3
1.2. Infrastructure Existante (déjà en place)	3
2. Architecture Déployée/Visée	4
2.1. Composants	4
2.2. Table d'adressage IP	4
2.3. Architecture Réseaux	5
2.4. Matrice de flux	5
3. Choix Techniques	6
3.1. Serveur SIEM : Virtualisation via OVA (TAAF-WAZUH-001)	6
3.2. Conteneurisation Docker (TAAF-DOCKER & TAAF-MONITORING)	6
3.3. Falco IDS	7
4. Instrutions d'installations	8
4.1. Falco	8
4.1.1. Installation de Falco	8
4.1.2. Vérification de l'installation	9
4.1.3. Configuration de FALco	9
4.1.4. Mise en place des règles Falco	10
4.2. Wazuh	14
4.2.1. Installation des agents Wazuh	14
4.2.2. Règles de corrélation Wazuh	15
4.2.3. Dashboard configuration	17
4.2.4. Intégration de Discord	18
4.3. Configuration de la Stack Monitoring (Grafana Alloy → Loki)	20
4.3.1. Configuration Grafana Alloy pour collecter les logs Falco	20
4.3.2. Redémarrage du service	20
4.4. Configuration du Pare-feu OPNsense et Agent Wazuh	21
4.4.1. Configuration des interfaces réseau	21
4.4.2. Mise en place des règles NAT / Port Forwarding	21
4.4.3. Configuration des règles ACL (Access Control Lists)	21
4.4.4. Installation et configuration de l'agent	23
5. Conclusion	24
5.1. Synthèse Technique	24

1. Introduction

1.1. Contexte

Ce document détaille l'implémentation de la couche de sécurité pour l'infrastructure des **Terres Australes et Antarctiques Françaises (TAAF)**.

Face aux besoins de traçabilité, de conformité et de protection des documents sensibles hébergés sur Nextcloud, nous déployons une solution **SIEM (Security Information and Event Management)** basée sur **Wazuh**. Cette solution permet la détection d'intrusion (HIDS), la surveillance de l'intégrité des fichiers (FIM) et la centralisation des logs.

1.2. Infrastructure Existante (déjà en place)

Il y a actuellement 2 machines:

- **TAAF-DOCKER-001 (Nextcloud + PostgreSQL)** : L'agent **surveille les conteneurs applicatifs, les tentatives d'accès** à la base de données et l'intégrité des fichiers stockés.
- **TAAF-MONITORING-001 (Stack Observabilité)** : L'agent **sécurise la stack de monitoring** (Caddy, Loki, Grafana, Prometheus, Alloy) pour **éviter toute compromission des outils** de surveillance.

2. Architecture Déployée/Visée

L'infrastructure est composée de trois machines virtuelles interconnectées sur un réseau privé.

2.1. Composants

L'architecture de sécurité s'articule autour de trois nœuds distincts :

- **Serveur Central SIEM (TAAF-WAZUH-001)** : Une nouvelle machine virtuelle dédiée qui héberge le « **cerveau** » du système (Manager, Indexer et Dashboard). Elle **centralise, analyse et stocke les alertes** de sécurité provenant de l'ensemble du parc.
- **Agents de Collecte Falco (sera déployés sur l'infrastructure existante)** : Des agents légers sont installés sur les serveurs de production pour **remonter la télémétrie et les logs en temps réel** vers le serveur central.
- **Sur TAAF-DOCKER-001 (Nextcloud + PostgreSQL + Falco IDS)** : L'agent **surveille les conteneurs applicatifs, les tentatives d'accès** à la base de données et l'intégrité des fichiers stockés.
- **Sur TAAF-MONITORING-001 (Stack Observabilité)** : L'agent **sécurise la stack de monitoring** (Caddy, Loki, Grafana, Prometheus, Alloy, Falco IDS) pour **éviter toute compromission des outils** de surveillance.

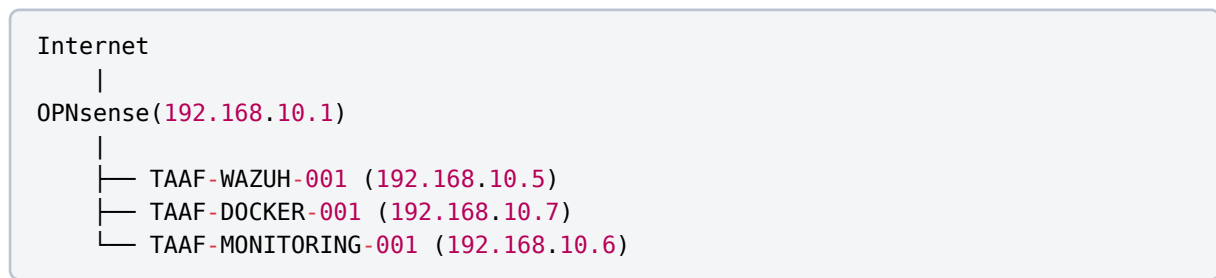
2.2. Table d'adressage IP

Machine	IP
TAAF-DOCKER-001	192.168.10.7
TAAF-MONITORING-001	192.168.10.6
WAZUH_MANAGER	192.168.10.5

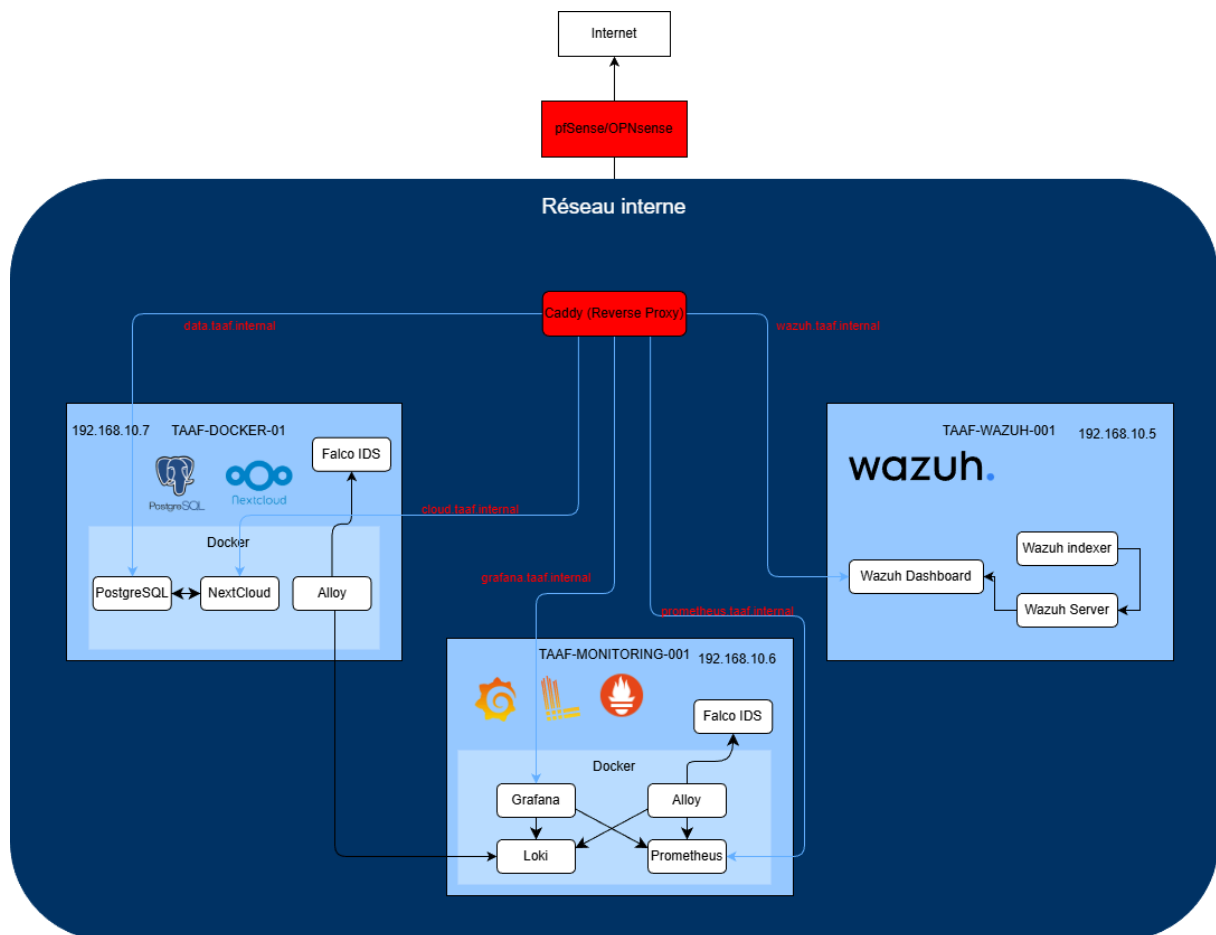
i Information

L'accès aux services est sécurisé via un pare-feu pfSense/OPNsense et un reverse proxy Caddy pour les domaines

2.3. Architecture Réseaux



2.4. Matrice de flux



3. Choix Techniques

3.1. Serveur SIEM : Virtualisation via OVA (TAAF-WAZUH-001)

Choix retenu : Déploiement de l'appliance virtuelle officielle (OVA) fournie par Wazuh.

Justification :

- **Stabilité et Performance :** L'OVA est une image pré-configurée par l'éditeur. Le système d'exploitation (basé sur AlmaLinux) est déjà optimisé pour supporter la charge d'indexation (réglages Kernel, limites de fichiers, etc.), ce qui élimine les risques d'erreurs humaines lors d'une installation manuelle.
- **Rapidité de mise en œuvre (Time-to-market) :** Contrairement à une installation « from scratch » ou Docker complexe à orchestrer pour la partie serveur, l'import de l'OVA permet d'avoir un SIEM opérationnel en quelques minutes.

Maintenance simplifiée : La mise à jour se fait globalement sur l'appliance, garantissant la cohérence entre le Manager, l'Indexer et le Dashboard.

3.2. Conteneurisation Docker (TAAF-DOCKER & TAAF-MONITORING)

Choix retenu : Docker et Docker Compose pour Nextcloud, PostgreSQL et la stack Monitoring.

Justification :

- **Isolation des services :** Chaque service tourne dans son propre conteneur avec ses propres bibliothèques. Cela évite les conflits de version (ex: Python pour l'un, PHP pour l'autre) sur la machine hôte.
- **Portabilité et Reproductibilité :** L'infrastructure est définie sous forme de code (docker-compose.yml). En cas de crash de la VM TAAF-DOCKER-001, il suffit de relancer le fichier Compose pour retrouver le service à l'identique.
- **Facilité de mise à jour :** Les mises à jour de Nextcloud ou Grafana se font simplement en changeant le tag de l'image Docker, sans risquer de casser la configuration du système d'exploitation sous-jacent.

3.3. Falco IDS

Choix retenu: Falco IDS sera installer nativement sur les machines TAAF-DOCKER-001 et TAAF-MONITORING-001

Justification:

- **Accès direct au kernel:** L'installation native permet à Falco d'accéder directement au kernel via eBPF ou un module kernel, garantissant une détection plus fiable des appels système.
- **Sécurité renforcée:** Installer Falco nativement réduit la surface d'attaque en évitant l'exécution d'un conteneur fortement privilégié.
- **Fiabilité au démarrage:** Falco installé sur l'hôte démarre avec le système et reste opérationnel même si Docker est indisponible.
- **Meilleures performances:** L'exécution native limite l'overhead lié à la conteneurisation et offre de meilleures performances en traitement des événements.
- **Simplicité d'administration:** L'installation native facilite la maintenance, la gestion des logs et le dépannage grâce à une intégration directe avec le système.

4. Instructions d'installations

4.1. Falco

Falco est un **IDS (Intrusion Detection System)** open-source conçu pour la **détection d'activités suspectes en temps réel** sur les systèmes Linux et les conteneurs Docker.

Falco agit comme un système de détection d'intrusion runtime qui :

- **Surveille les appels système** : Capture toutes les actions effectuées par les processus et conteneurs (création de fichiers, connexions réseau, exécution de commandes, etc.)
- **Détecte les anomalies** : Compare les événements observés avec des règles de sécurité prédéfinies
- **Alerte en temps réel** : Génère des événements de sécurité lorsqu'un comportement suspect est détecté
- **Pas de blocage** : Falco est un IDS passif (détection uniquement), pas un IPS (Intrusion Prevention System qui bloque les actions)
- **Logs vers fichier** : Les événements sont écrits dans `/var/log/falco.log` pour collecte par Grafana Alloy

4.1.1. Installation de Falco

```
# Sur la VM TAAF-DOCKER-001 et TAAF-MONITORING-001
# Vérifier que Falco n'est pas déjà installé

which falco

# Ajouter la clé GPG de Falco

curl -fsSL https://falco.org/repo/falcosecurity-packages.asc | \ sudo gpg --
dearmor -o /usr/share/keyrings/falco-archive-keyring.gpg

# Ajouter le dépôt Falco

echo "deb [signed-by=/usr/share/keyrings/falco-archive-keyring.gpg] https://
download.falco.org/packages/deb stable main" | sudo tee /etc/apt/sources.list.d/
falcosecurity.list

# Mettre à jour la liste des paquets

sudo apt update

# Installer Falco
```

```
sudo apt install -y linux-headers-$(uname -r)
sudo apt install -y falco
```

4.1.2. Vérification de l'installation

```
# Vérifier que Falco est installé

falco --version

# Vérifier le service Falco

sudo systemctl status falco

# Démarrer Falco si nécessaire

sudo systemctl enable falco
sudo systemctl start falco
```

4.1.3. Configuration de Falco

i Information

Le fichier de configuration de falco se trouve dans « **/etc/falco/falco.yml** »

```
# Éditer la configuration Falco
sudo nano /etc/falco/falco.yml

# Vérifier/Modifier les paramètres suivants :
# json_output et json_include_output_property force Falco à structurer ses
# alertes au format JSON plutôt qu'en texte brut
json_output: true
json_include_output_property: true
# log_level: info définit la verbosité des journaux internes de Falco.
log_level: info

#file_output permet la configuration de la sortie vers un fichier physique sur le
#disque
file_output:
  enabled: true # Active l'écriture des alertes dans un fichier
  keep_alive: false # Indique à Falco de fermer le fichier après chaque écriture
  # d'alerte
  filename: /var/log/falco.log # Spécifie le chemin où les alertes seront
  # enregistrées

# Redémarrer Falco pour appliquer les changements
sudo systemctl restart falco
```

```
# Vérifier les logs  
sudo tail -f /var/log/falco.log
```

4.1.4. Mise en place des règles Falco

i Information

Ne modifiez JAMAIS les fichiers `falco_rules.yaml` (règles par défaut) Créez vos règles personnalisées dans `/etc/falco/rules.d/...`

Les seront créés sur la machine TAAF-DOCKER-001, dans:

- `/etc/falco/rules.d/taaf-postgres-rules.yaml`:

```
- rule: Shell in PostgreSQL Container  
  desc: Détection d'accès shell dans le conteneur PostgreSQL  
  condition: >  
    evt.type = execve and  
    container.name contains "postgres-nextcloud-af" and  
    proc.name in (sh, bash, zsh, csh, fish)  
  output: >  
    Shell détecté dans PostgreSQL (user=%user.name proc=%proc.name  
    container=%container.name cmd=%proc.cmdline)  
  priority: CRITICAL  
  tags: [database, shell, intrusion]  
  
- rule: PostgreSQL Config Access  
  desc: Accès aux fichiers de configuration PostgreSQL  
  condition: >  
    (evt.type in (open, openat, openat2) and evt.is_open_read=true) and  
    container.name contains "postgres-nextcloud-af" and  
    fd.name contains "postgresql.conf"  
  output: >  
    Accès config PostgreSQL (user=%user.name file=%fd.name  
    container=%container.name)  
  priority: WARNING  
  tags: [database, config, file_access]
```

- `/etc/falco/rules.d/taaf-nextcloud-rules.yaml`:

```
- rule: Nextcloud Folder Creation  
  desc: Détection de création de dossier dans Nextcloud  
  condition: >  
    evt.type in (mkdir, mkdirat) and  
    container.name contains "nextcloud" and  
    evt.arg.path startswith "/var/www/html/data/"  
  output: >  
    Création de dossier Nextcloud détectée (user=%user.name folder=%evt.arg.path  
    container=%container.name cmd=%proc.cmdline)  
  priority: ERROR
```

```
tags: [nextcloud, folder, creation, file_operation]

- rule: Nextcloud Dangerous Command Detection
  desc: Détecte l'exécution manuelle de rm -rf sur les données
  condition: >
    evt.type = execve and
    container.name contains "nextcloud" and
    proc.name = rm and
    (proc.args contains "r" or proc.args contains "f") and
    proc.cmdline contains "/var/www/html/data/"
  output: >
    Suppression de dossier Nextcloud détectée (user=%user.name
command=%proc.cmdline
    container=%container.name)
  priority: CRITICAL
  tags: [nextcloud, security, command]

- rule: Nextcloud Folder Deletion (Result)
  desc: Détection de suppression de dossier dans Nextcloud (rmdir syscall)
  condition: >
    evt.type = rmdir and
    container.name contains "nextcloud" and
    evt.arg.path startswith "/var/www/html/data/"
  output: >
    Suppression de dossier Nextcloud détectée (user=%user.name
folder=%evt.arg.path
    container=%container.name cmd=%proc.cmdline)
  priority: WARNING
  tags: [nextcloud, folder, deletion, file_operation]

- rule: Nextcloud Config Modification
  desc: Modification de la configuration Nextcloud
  condition: >
    (evt.type in (open, openat, openat2) and evt.is_open_write=true) and
    container.name contains "nextcloud" and
    fd.name contains "config.php"
  output: >
    Modification config Nextcloud (user=%user.name file=%fd.name
    container=%container.name proc=%proc.name cmd=%proc.cmdline)
  priority: CRITICAL
  tags: [nextcloud, config, file_modification]
```

- /etc/falco/rules.d/taaf-postgres-rules.yaml:

```
- rule: Shell in PostgreSQL Container
  desc: Détection d'accès shell dans le conteneur PostgreSQL
  condition: >
    evt.type = execve and # S'active quand un nouveau
programme est lancé (exécution)
    container.name contains "postgres-nextcloud-af" and # Filtre : Uniquement si
cela se passe dans ton conteneur DB
    proc.name in (sh, bash, zsh, csh, fish) # Filtre : Si le programme lancé
est un interpréteur de commandes (Shell)
```

```
output: >
  Shell détecté dans PostgreSQL (user=%user.name proc=%proc.name
  container=%container.name cmd=%proc.cmdline) # Message d'alerte avec les
détails (qui, quoi, où)
  priority: CRITICAL # Niveau d'alerte maximum
(c'est anormal sur une DB)
  tags: [database, shell, intrusion]

- rule: PostgreSQL Config Access
  desc: Accès aux fichiers de configuration PostgreSQL
  condition: >
    (evt.type in (open, openat, openat2) and evt.is_open_read=true) and #
S'active quand un fichier est OUVERT en LECTURE
    container.name contains "postgres-nextcloud-af" and #
Uniquement dans le conteneur DB
    fd.name contains "postgresql.conf" #
Uniquement si le nom du fichier contient "postgresql.conf"
  output: >
    Accès config PostgreSQL (user=%user.name file=%fd.name
    container=%container.name)
  priority: WARNING # Niveau moyen (peut être une
surveillance normale ou du vol de config)
  tags: [database, config, file_access]
```

• /etc/falco/rules.d/taaf-system-rules.yaml:

```
- rule: Container Privilege Escalation (Sudo/Su)
  desc: Détection de sudo ou su à l'intérieur du conteneur Nextcloud
  condition: >
    spawned_process and # Macro Falco : raccourci pour
"un processus démarre" (evt.type=execve)
    (proc.name = "sudo" or proc.name = "su") and # Si la commande est
'sudo' (admin) ou 'su' (changement d'user)
    container.name contains "nextcloud" # Uniquement dans le conteneur
Nextcloud
  output: >
    ESCALADE DE PRIVILEGES DANS CONTENEUR (user=%user.name cmd=%proc.cmdline
    container=%container.name)
  priority: CRITICAL # Critique car un conteneur ne
doit pas utiliser sudo
  tags: [security, container, mitre_privilege_escalation]

- rule: Sensitive File Access
  desc: Accès à des fichiers sensibles du système
  condition: >
    (evt.type in (open, openat, openat2) and evt.is_open_read=true) and #
Ouverture de fichier en LECTURE
    fd.name in (/etc/shadow, /etc/sudoers) # Si le
fichier est la liste des mots de passe (shadow) ou des droits admin (sudoers)
  output: >
    Accès fichier sensible (file=%fd.name user=%user.name proc=%proc.name
    container=%container.name)
  priority: WARNING
  tags: [file_access, sensitive]
```

- /etc/falco/rules.d/container_shell.yaml:

```
- rule: Shell in Critical Container
  desc: Détecte l'exécution d'un shell (bash/sh) dans un conteneur
  condition: >
    evt.type = execve and          # Un processus démarre
    container.id != host and      # Le processus n'est PAS sur la machine hôte
    (donc il est dans un conteneur Docker)
    proc.name in (bash, sh) and   # Le programme est un terminal standard
    proc.tty != 0                 # Il y a un terminal interactif (TTY) attaché :
    un humain est derrière le clavier !
  output: >
    Shell access detected in container (container=%container.name user=%user.name
    command=%proc.cmdline)
  priority: WARNING
  tags: [container, security, shell, mitre_execution]
```

- /etc/falco/rules.d/nextcloud_data.yaml

```
- rule: Nextcloud Folder Creation
  desc: Détection de création de dossier dans Nextcloud
  condition: >
    evt.type in (mkdir, mkdirat) and          # S'active sur les appels
    système de création de dossier (Make Directory)
    container.name contains "nextcloud" and    # Cible le conteneur
    Nextcloud
    evt.arg.path startswith "/var/www/html/data/" # Uniquement si le chemin
    commence par le dossier des données utilisateur
  output: >
    Création de dossier Nextcloud détectée (user=%user.name folder=%evt.arg.path
    container=%container.name cmd=%proc.cmdline)
  priority: ERROR                             # Niveau élevé (utile pour
  l'audit)
  tags: [nextcloud, folder, creation, file_operation]

- rule: Nextcloud Dangerous Command Detection
  desc: Détecte l'exécution manuelle de rm -rf sur les données
  condition: >
    evt.type = execve and          # Un processus démarre
    container.name contains "nextcloud" and    # Cible Nextcloud
    proc.name = rm and             # La commande est
    'rm' (remove)
    (proc.args contains "r" or proc.args contains "f") and # Les arguments
    contiennent 'r' (récuratif) ou 'f' (force)
    proc.cmdline contains "/var/www/html/data/" # La commande tape dans le
    dossier des données
  output: >
    Suppression de dossier Nextcloud détectée (user=%user.name
    command=%proc.cmdline
    container=%container.name)
  priority: CRITICAL                       # Critique : C'est une
  tentative de destruction massive
  tags: [nextcloud, security, command]
```

```
- rule: Nextcloud Folder Deletion (Result)
  desc: Détection de suppression de dossier dans Nextcloud (rmdir syscall)
  condition: >
    evt.type = rmdir and                                # Appel système "Remove
Directory" (le résultat technique de la commande rm)
    container.name contains "nextcloud" and
    evt.arg.path startswith "/var/www/html/data/"      # Le dossier supprimé était
dans /data/
  output: >
    Suppression de dossier Nextcloud détectée (user=%user.name
folder=%evt.arg.path
    container=%container.name cmd=%proc.cmdline)
  priority: WARNING
  tags: [nextcloud, folder, deletion, file_operation]
```

- /etc/falco/rules.d/nextcloud_config.yaml:

```
- rule: Nextcloud Config Modification
  desc: Modification de la configuration Nextcloud
  condition: >
    (evt.type in (open, openat, openat2) and evt.is_open_write=true) and #
Ouverture de fichier en ÉCRITURE (modification)
    container.name contains "nextcloud" and
    fd.name contains "config.php"                                # Le fichier visé est le cœur
de la config Nextcloud
  output: >
    Modification config Nextcloud (user=%user.name file=%fd.name
    container=%container.name proc=%proc.name cmd=%proc.cmdline)
  priority: CRITICAL                                           # Critique : Si on change ce
fichier, on peut casser le site ou voler les accès DB
  tags: [nextcloud, config, file_modification]
```

4.2. Wazuh

4.2.1. Installation des agents Wazuh

i Information

La méthode d'installations est la même pour TAAF-DOCKER-001 et TAAF-MONITORING-001

Nous avons suivis le guide d'installation de la documentation Wazuh (le lien de la doc: <https://documentation.wazuh.com/current/installation-guide/wazuh-agent/wazuh-agent-package-linux.html>)

```
# Installation des paquets manquants
apt-get install gnupg apt-transport-https

# Installation de la clé GPG
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

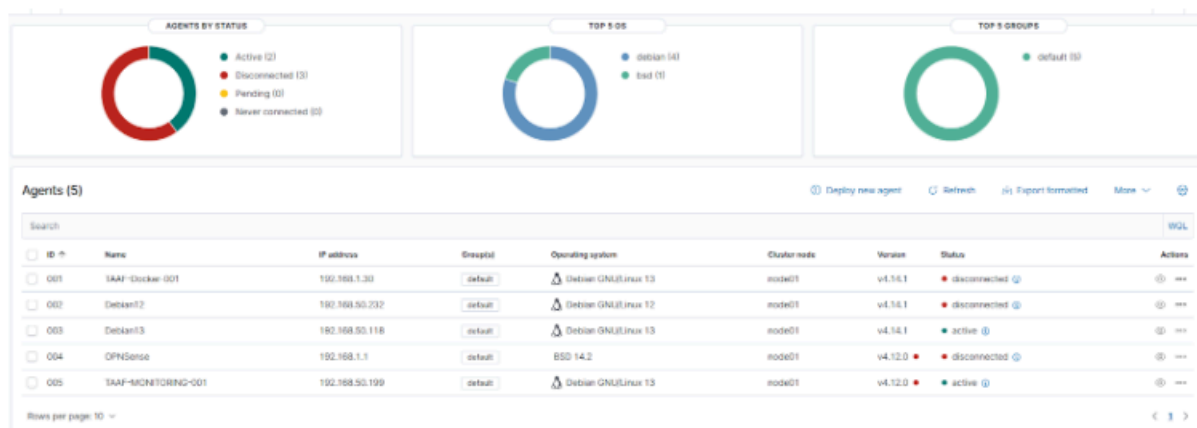
# Ajouter le repository
echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.
x/apt/ stable main" | tee -a /etc/apt/sources.list.d/wazuh.list

#Mets à jour
apt-get update

#Déploiement de l'agent
WAZUH_MANAGER="192.168.10.5" apt-get install wazuh-agent

# Activer et lancer wazuh agent
systemctl daemon-reload
systemctl enable wazuh-agent
systemctl start wazuh-agent
```

Nous voyons toutes nos machines qui sont remonter dans le dashboard:



4.2.2. Règles de corrélation Wazuh

Un fichier de règles personnalisées a été créé pour détecter des comportements suspects spécifiques, notamment via l'intégration avec Falco.

Fichier de configuration : /var/ossec/etc/rules/100-taaf-custom.xml Les règles suivantes ont été définies dans le groupe taaf, falco, custom:

- Règle 100001 (Niveau 10) : Tentative d'intrusion via shell
 - Déclencheur : Détection par Falco d'une tentative d'accès shell.

- Condition : Le champ output contient « Shell ».
- Groupe : intrusion_attempt, pci_dss_10.6.1.
- Règle 100002 (Niveau 12) : Multiples tentatives d'accès shell
 - Description : Corrélation de multiples tentatives (3 tentatives en 5 minutes).
 - Condition : Si la règle 100001 est déclenchée 3 fois dans un laps de temps de 300 secondes.
 - Groupe : intrusion_attempt, attack, pci_dss_11.4.
- Règle 100003 (Niveau 8) : Modification de fichiers de configuration
 - Déclencheur : Modification détectée sur des fichiers .conf ou config.php.
 - Groupe : file_integrity, pci_dss_11.5.
 - Exemple d'alerte observée : Suppression de dossier Nextcloud ou ouverture de /etc/shadow par un programme non fiable.
- Règle 100004 (Niveau 12) : Création de webshell suspicieux
 - Déclencheur : Création de fichiers avec extensions .php, .jsp, ou .asp.
 - Condition spécifique : Write below rpm database.
 - Groupe : web_attack, attack

```
<group name="taaf,falco,custom">
<!-- Règle 1 : Tentative d'intrusion via shell -->
<rule id="100001" level="10">
<if_group>falco</if_group>
<field name="output">Shell</field>
<description>Falco: Tentative d'accès shell détectée</description>
<group>intrusion_attempt,pci_dss_10.6.1,</group>
</rule>
<!-- Règle 2 : Multiples tentatives (corrélation) -->
<rule id="100002" level="12" frequency="3" timeframe="300">

<if_matched_sid>100001</if_matched_sid>
<description>Falco: Multiples tentatives d'accès shell (3 en 5
min)</description>
<group>intrusion_attempt,attack,pci_dss_11.4,</group>
</rule>
<!-- Règle 3 : Modification de fichiers de configuration -->
<rule id="100003" level="8">
<if_group>falco</if_group>
<match>\.conf|config\.php</match>
<description>Falco: Modification de fichier de configuration
détectée</description>
<group>file_integrity,pci_dss_11.5,</group>
</rule>
<!-- Règle 4 : Création de webshell suspicieux -->
<rule id="100004" level="12">
<if_group>falco</if_group>
<match>\.php|\.jsp|\.asp</match>
<field name="output">Write below rpm database</field>
<description>Falco: Création potentielle de webshell</description>
<group>web_attack,attack,</group>
</rule>
</group>
```

Puis nous redémarrons le service pour appliquer les règles:

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

4.2.3. Dashboard configuration

Procédure de création :

- Accéder à la création de dashboard
 - Connectez-vous au Wazuh Dashboard : <https://192.168.10.5>
 - Menu hamburger (☰) → Dashboards Management → Create Dashboard

Visualisation 1 : Événements par priorité

- Type : Pie Chart
- Index pattern : wazuh-alerts-*
- Metrics : Count
- Buckets : Split slices → Aggregation: Terms → Field: rule.level
- Save as : « Événements par priorité »

Visualisation 2 : Timeline des alertes

- Type : Line Chart
- Index : wazuh-alerts-*
- X-axis : Date Histogram → Field: @timestamp → Interval: 1h
- Y-axis : Count
- Save as : « Timeline des alertes »

Visualisation 3 : Top 10 règles déclenchées

- Type : Data Table
- Index : wazuh-alerts-*
- Metrics : Count
- Buckets :
 - Split rows → Terms → rule.description.keyword → Size: 10
 - Split rows → Terms → rule.id → Size: 10
- Save as : « Top 10 règles »

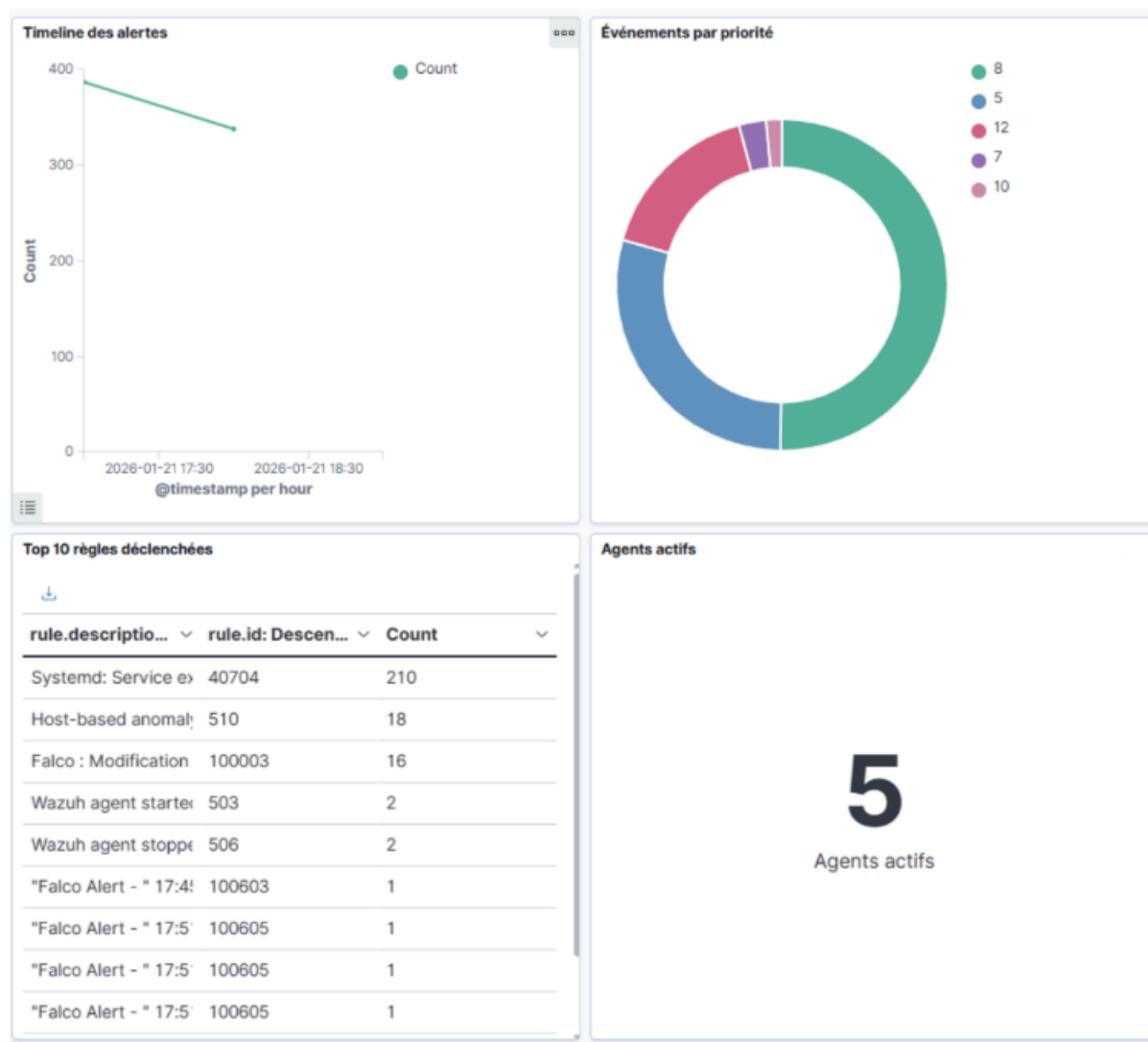
Visualisation 4 : Agents actifs

- Type : Metric
- Index : wazuh-monitoring-
- Metrics : Unique Count → Field: agent.id
- Save as : « Agents actifs »

Assembler le dashboard:

- Create Dashboard → « TAAF Security Dashboard »
- Add → Sélectionner vos 4 visualisations
- Organiser et redimensionner
- Save

Résultat:



4.2.4. Intégration de Discord

4.2.4.1. Création du Webhooks:

- Créer un nouveau salon
- Cliquez sur l'icône de rouage (Paramètres du salon) à côté du nom du salon.
- Dans le menu de gauche, cliquez sur l'onglet Intégrations.
- Cliquez sur le bouton Créer un webhook

4.2.4.2. Configuration du Wazuh Manager

Modification de la configuration (ossec.conf) :

Ajout du bloc d'intégration pour activer le script custom-discord au niveau d'alerte 8.

```
<integration>
<name>custom-discord</name>
<hook_url>https://discord.com/api/webhooks/API_KEY</hook_url>
<level>8</level>
```

```
<alert_format>json</alert_format>
</integration>
```

Puis nous redémarrons le service pour appliquer les règles:

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

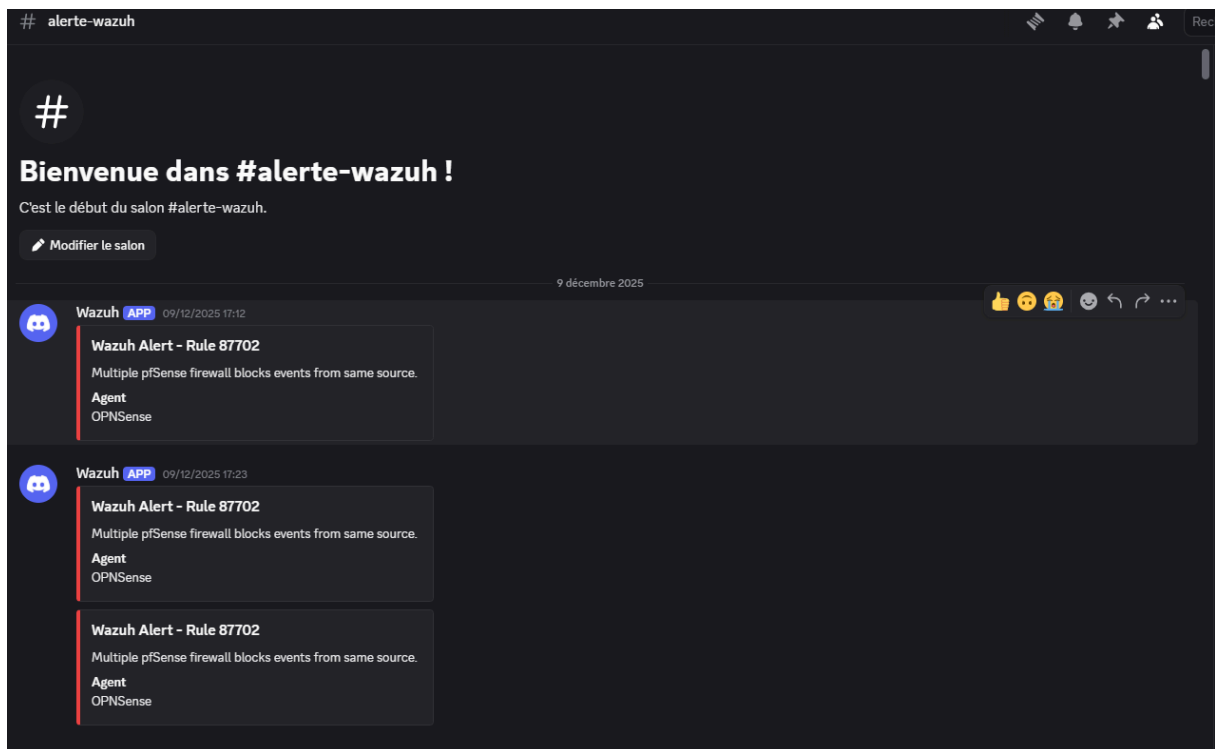
Installation des scripts d'intégration : Téléchargement des scripts custom-discord et custom-discord.py depuis le dépôt GitHub maikroservice/wazuh-integrations dans le dossier /var/ossec/integrations.

```
cd /var/ossec/integrations

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
#Permissions et dépendances
sudo chmod 750 /var/ossec/integrations/custom-*
sudo chown root:wazuh /var/ossec/integrations/custom-*

#Redémarrage du service via /var/ossec/bin/wazuh-control
/var/ossec/bin/wazuh-control restart
```

Les alertes sont désormais visibles directement dans le salon Discord



4.3. Configuration de la Stack Monitoring (Grafana Alloy → Loki)

4.3.1. Configuration Grafana Alloy pour collecter les logs Falco

i Information

La base de notre fichier de configuration alloy.alloy se trouve sur GitHub fournie par M.Fontaine Thibaut. Nous allons juste montrer les modifications faites pour notre environnement.

Sur TAAF-MONITORING-001, voici les modifications faites du fichier de configuration alloy.alloy:

```
// =====  
// FALCO IDS - DÉTECTION D'INTRUSION  
// =====  
local.file_match "falco_logs" {  
  path_targets = [  
    {  
      "__path__"    = "/var/log/falco_events.json",  
      "job"         = "falco-ids",  
      "vm"          = "taaf-monitoring-001",  
      "service"     = "falco",  
      "location"    = "Crozet",  
      "log_type"    = "security",  
      "criticality" = "critical",  
      "mission"     = "intrusion-detection",  
    },  
  ],  
}  
  
// =====  
// LOKI WRITE - ENVOI VERS LOKI (SIMPLIFIÉ)  
// =====  
  
loki.write "default" {  
  endpoint {  
    url = "http://192.168.50.199:3100/loki/api/v1/push"  
  }  
}
```

4.3.2. Redémarrage du service

```
# Redémarrer Grafana Alloy et Loki  
docker compose down alloy loki  
docker compose up -d loki alloy
```

```
# Vérifier les logs Alloy
docker compose logs alloy
```

4.4. Configuration du Pare-feu OPNsense et Agent Wazuh

4.4.1. Configuration des interfaces réseau

Nous avons configuré le pare-feu OPNsense avec deux interfaces réseau distinctes afin de segmenter correctement les flux:

- **WAN** : interface connectée au réseau externe / Internet.
- **LAN** : interface connectée au réseau privé, sur lequel sont déployées l'ensemble des machines virtuelles (serveur Wazuh, serveur de supervision, serveur Docker, etc.).

Cette séparation permet de garantir une isolation entre le réseau interne et le réseau externe, conformément aux bonnes pratiques de sécurité.

4.4.2. Mise en place des règles NAT / Port Forwarding

Afin de permettre l'accès aux services hébergés sur les machines virtuelles depuis le réseau RT, nous avons configuré plusieurs règles de redirection de ports (NAT) sur OPNsense.

Interface	Proto	Source	Dest. Adresse	Dest. Ports	NAT IP	NAT Ports
LAN	TCP	LAN net	LAN adresse	443 (HTTPS)	192.168.10.5	443 (HTTPS)
LAN	TCP	LAN net	LAN adresse	3000 (HBCI)	192.168.10.6	3000 (HBCI)
LAN	TCP	LAN net	LAN adresse	8080	192.168.10.7	8081

Règle 1: Nous avons configuré une règle NAT permettant l'accès au tableau de bord Wazuh depuis le réseau RT.

Règle 2: Une règle NAT a été mise en place afin de permettre l'accès à Grafana hébergé sur le serveur de monitoring.

Règle 3: Nous avons également configuré une règle NAT pour permettre l'accès au service Nextcloud hébergé sur la VM TAAF-DOCKER-001.

4.4.3. Configuration des règles ACL (Access Control Lists)

Les règles de filtrage ont été configurées de manière restrictive afin de garantir un haut niveau de sécurité.

4.4.3.1. ACL 1 — Autorisation d'accès aux services depuis le réseau RT

Nous avons autorisé l'accès aux services uniquement depuis le réseau RT via le protocole IPv4 TCP sur l'alias ACL1Port.

<input type="checkbox"/>		IPv4 TCP	LAN net	*	IP_VM_MONITORING, IP_VM_WAZUH, IP_VM_DOCKER	ACL1Port	*	*	Accès services SAE depuis LAN
--------------------------	--	----------	---------	---	---------------------------------------------	----------	---	---	-------------------------------

4.4.3.2. ACL 2 — Blocage des accès depuis Internet

Nous avons explicitement bloqué tout accès aux services internes (VM Monitoring, Wazuh, Docker) depuis l'interface WAN afin d'empêcher toute exposition directe sur Internet.

Pare-feu: Règles: WAN									
Sélectionnez une catégorie									
Inspecter									
<input type="checkbox"/>	Protocole	Source	Port	Destination	Port	Passerelle	Planifier	Description	
Règles générées automatiquement									
<input type="checkbox"/>	IPv4	*	*	*	*	*	*		
<input type="checkbox"/>	IPv4	*	*	IP_VM_MONITORING, IP_VM_WAZUH, IP_VM_DOCKER	*	*	*	Bloquer accès Internet aux services SAE	

4.4.3.3. ACL 3 — Autorisation des communications inter-machines virtuelles

Nous avons autorisé les flux nécessaires au bon fonctionnement des services entre les machines virtuelles internes[cite: 88, 89, 90]:

- **VM Docker vers Wazuh** : Flux IPv4 TCP/UDP vers 192.168.10.5.

<input type="checkbox"/>		IPv4 TCP/UDP	IP_VM_DOCKER	*	IP_VM_WAZUH	ACL3Port	*	*	Communication VM Docker vers Wazuh
--------------------------	--	--------------	--------------	---	-------------	----------	---	---	------------------------------------

- **VM Docker vers supervision** : Flux IPv4 TCP vers 192.168.10.6 (Loki, port 3100).

<input type="checkbox"/>		IPv4 TCP	IP_VM_DOCKER	*	IP_VM_MONITORING	3100	*	*	Communication VM Docker vers Loki
--------------------------	--	----------	--------------	---	------------------	------	---	---	-----------------------------------

4.4.3.4. ACL 4 — Politique de sécurité par défaut (Deny All)

Enfin, nous avons mis en place une règle de blocage globale afin de refuser tout trafic non explicitement autorisé.

<input type="checkbox"/>		IPv4	*	*	*	*	*	*	Deny all - Sécurité par défaut
--------------------------	--	------	---	---	---	---	---	---	--------------------------------

4.4.4. Installation et configuration de l'agent

4.4.4.1. Installation de l'agent sur OPNsense

Nous installons le paquet `os-wazuh-agent` sur OPNsense.

4.4.4.2. Configuration de l'agent

L'agent est configuré avec les informations du serveur de gestion.

Paramètre	Valeur configurée
Activer	Coché
Nom d'hôte du gestionnaire	192.168.10.5
Applications	audit (audit), configd (configd.py), dhcpd (dhcpd)
Événements de détection d'Intrusion	Coché
Réponse active : Activer	Coché
Ignorer les commandes du pare-feu	Aucun
Enrollment port	1514

5. Conclusion

L'implémentation de la solution Wazuh au sein de l'infrastructure des TAAF **répond de manière concrète aux exigences de sécurité** posées par l'utilisation de Nextcloud. En centralisant la gestion des logs et en automatisant la surveillance de l'intégrité des fichiers (FIM), nous avons transformé une infrastructure statique en un environnement **proactif face aux menaces cyber**.

5.1. Synthèse Technique

La mise en place du SIEM a permis de valider **trois piliers essentiels** :

- **La Visibilité** : Une vue d'ensemble en temps réel sur les accès aux documents sensibles.
- **La Réactivité** : La capacité du système HIDS à identifier et alerter sur des comportements suspects.
- **La Conformité** : Le respect des standards de traçabilité indispensables pour une administration publique.

Hoareau Anthony Bernard Didier Phamsi Lillian Pasqua Xavier

40 Av. De Soweto, Saint-Pierre 97410, La Réunion



**MINISTÈRE
DE L'ENSEIGNEMENT
SUPÉRIEUR,
DE LA RECHERCHE
ET DE L'ESPACE**

*Liberté
Égalité
Fraternité*