RAPPORT TP Wazuh R 5.CYBER.11

Université de la réunion / IUT

Département Réseaux, Télécommunication en Cybersécurité – 3ème années

R 5.CYBER.11

SUPERVISION DE LA SÉCURITÉ

EMMANUEL GRONDIN

Yann BOISVILLIERS

Objectifs

- Concevoir une architecture de supervision réseau et système.
- Configurer SNMP, agents et sondes de supervision.
- Collecter et centraliser les logs pour analyse.
- Corréler les événements dans un SIEM (Wazuh).
- Détecter un incident simulé (attaque brute-force SSH).
- Mise en place d'une gestion des événements basés sur ITIL et ITSM

Plan d'adressage IP

Machine	Rôle	OS	IP	Services / Agents
srv-linux	Serveur applicatif	Debian 12	192.168.1.65	Apache + Zabbix Agent + Wazuh Agent
srv-windows	Serveur applicatif	Windows Server 2016/2019	192.168.1.66	Zabbix Agent + Wazuh Agent
firewall	Pare-feu / routeur	pfSense / OPNsense	192.168.1.1	SNMP + Wazuh Agent
srv-monitoring	Supervision + SOC	Debian 12	192.168.1.30	Zabbix Server + Wazuh Manager
srv-glpi	ITSM / ticketing	Debian 12 (LAMP)	192.168.1.20	GLPI + Apache + MySQL

---

1. Introduction

La gestion d'une infrastructure informatique moderne repose sur deux piliers complémentaires : le Maintien en Condition Opérationnelle (MCO) et la cybersécurité. Dans un environnement hétérogène mêlant Linux, Windows et des équipements réseau, il est impossible pour un administrateur de surveiller manuellement l'état de chaque machine.

Ce projet (SAÉ 5.CYBER.03) a pour objectif de concevoir et déployer une architecture centralisée capable de répondre à ce besoin d'observabilité totale. Notre approche se divise en trois axes stratégiques :

1. **La Supervision (NOC) :** Avec **Zabbix**, pour surveiller la santé des serveurs (CPU, RAM, Disque) et la disponibilité des services critiques.
2. **La Sécurité (SOC) :** Avec le SIEM **Wazuh**, pour collecter les journaux (logs), analyser les comportements suspects et détecter les intrusions.
3. **L'Automatisation (ITSM) :** Avec **GLPI**, pour transformer les alertes techniques en tickets d'incidents, appliquant ainsi les bonnes pratiques ITIL pour une gestion structurée des problèmes.

Ce rapport détaille l'installation, la configuration et l'interconnexion de ces outils, jusqu'à la simulation d'incidents réels (disparition de fichiers critiques, attaques Brute-Force) pour valider l'automatisation de la chaîne de réaction.

1.1 Justification du choix des solutions (Zabbix vs Centreon)

Pour ce projet, le choix de **Zabbix** comme outil de supervision, au détriment de solutions comme Centreon, repose sur plusieurs critères stratégiques liés à notre infrastructure :

- **Performance des agents :** Zabbix utilise des agents natifs (Zabbix Agent) extrêmement légers et performants pour Windows et Linux. Contrairement à Centreon qui repose souvent sur le protocole SNMP (parfois lourd à configurer sur Windows), l'agent Zabbix permet une remontée de métriques beaucoup plus fine et en temps réel (Active Checks).
- **Flexibilité des déclencheurs (Triggers) :** Zabbix offre une gestion des seuils d'alerte beaucoup plus granulaire. Cela a été décisif pour notre scénario de surveillance de fichier témoin, où la réactivité devait être quasi instantanée.
- **Modernité et Intégration :** L'interface de Zabbix est plus intuitive pour la gestion de tableaux de bord personnalisés. De plus, sa structure s'intègre mieux dans une démarche de "Supervision as Code", facilitant ainsi la corrélation future avec notre SOC Wazuh.

---

2. Architecture globale

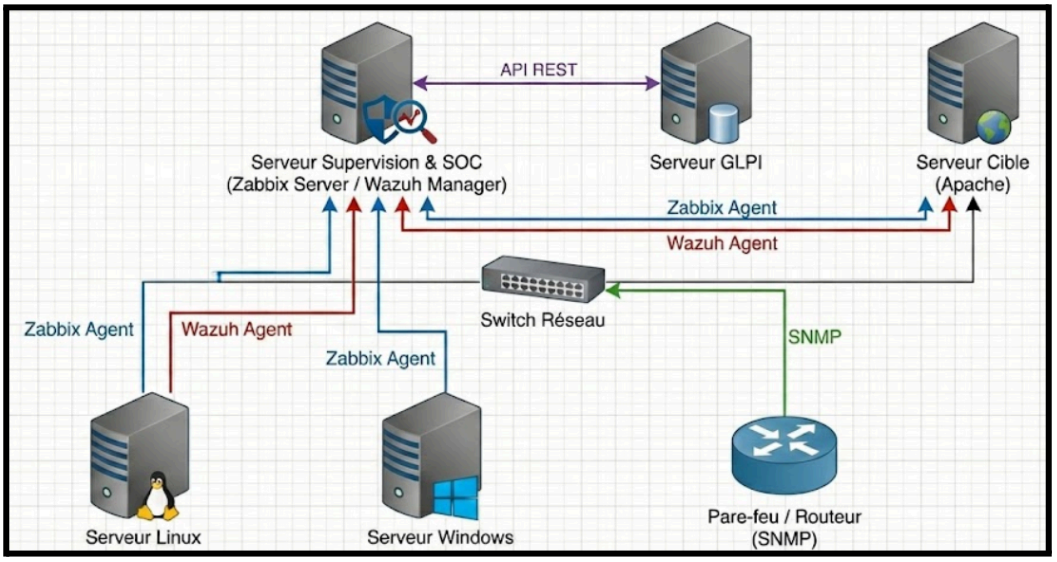
2.1 Schéma de l’architecture

L’architecture repose sur trois serveurs principaux :

- Un serveur de **supervision et SOC**
- Un serveur **GLPI**
- Un serveur **Apache** servant de machine cible

Les flux suivants sont utilisés :

- Zabbix Agent → Zabbix Server
- Agents Wazuh → Wazuh Manager
- API REST → GLPI
- **Composants réseau** : serveur Linux, serveur Windows, pare-feu SNMP



Configuration de la supervision sur le pare-feu OPNsense

Dans le cadre de l'architecture de supervision, le pare-feu OPNsense joue un rôle critique. Nous avons mis en place une double stratégie de surveillance : via le protocole standard **SNMP** et via l'**agent Zabbix** natif pour une remontée d'informations plus fine.

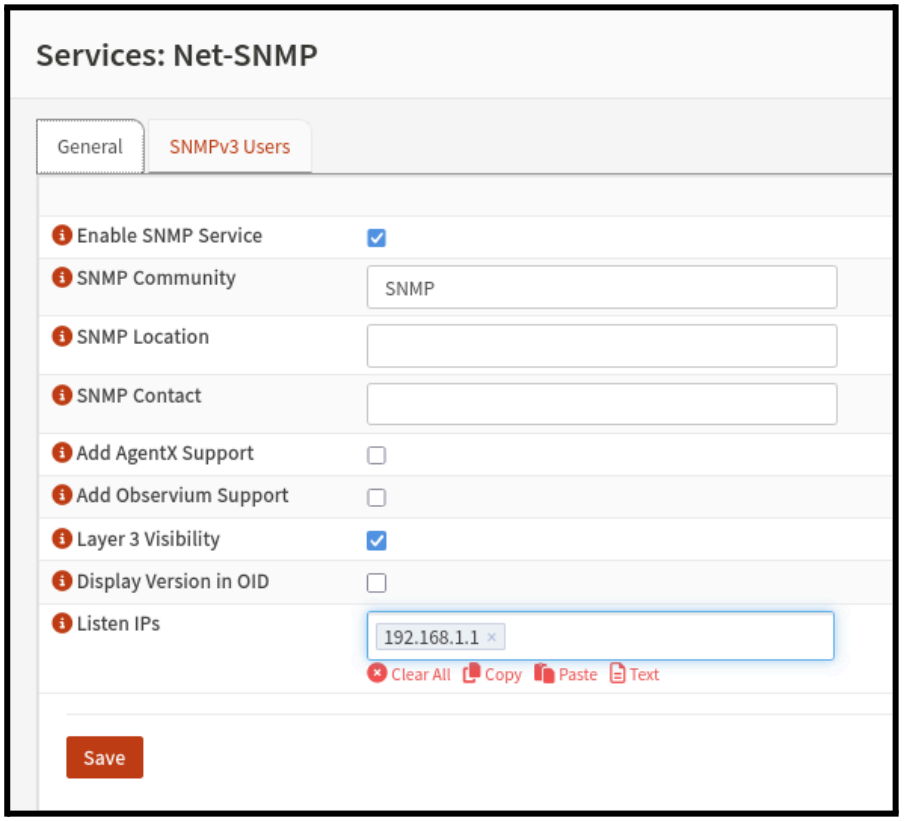
Note : L'adressage IP utilisé dans cette mise en œuvre (192.168.1.x) a été adapté par rapport au sujet initial pour correspondre à l'environnement de test.

---

3 .Configuration du service Net-SNMP sur OPNSENSE

Afin de permettre une supervision standardisée (état des interfaces, bande passante, charge CPU), j'ai activé et configuré le service SNMP v2c.

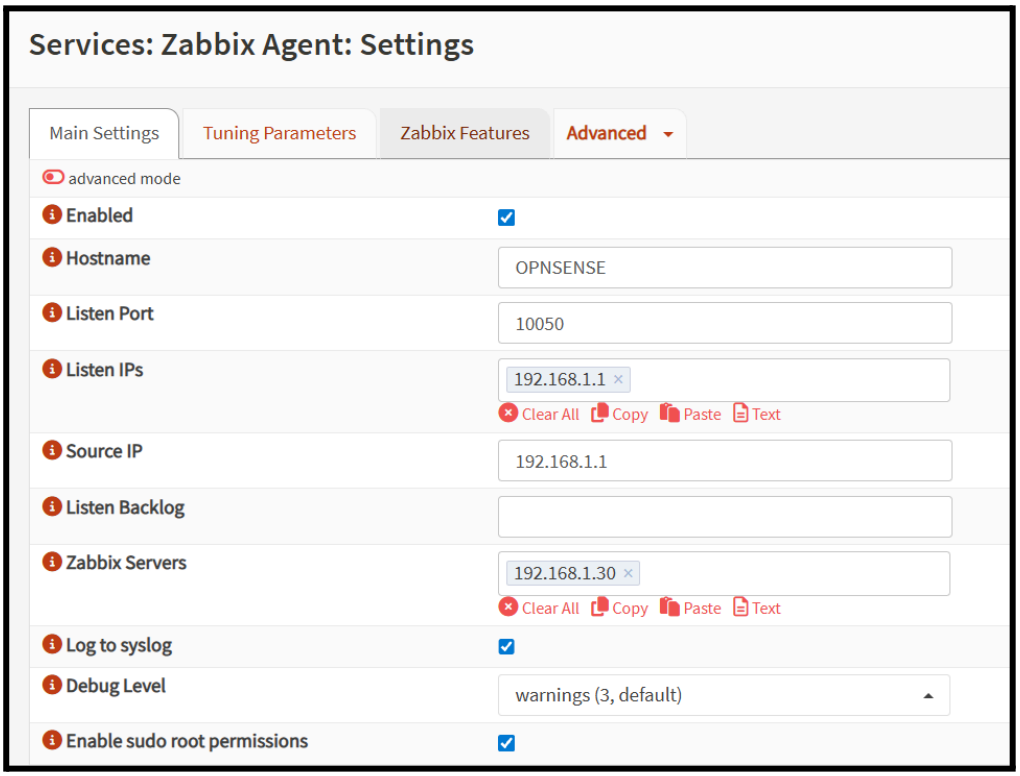
- **Action :** Activation du service dans *Services Net-SNMP*.
- **Communauté SNMP :** Définie sur SNMP. C'est le mot de passe qui permettra au serveur de supervision de l'interroger.
- **Interface d'écoute :** Le service écoute sur l'interface LAN (192.168.1.1) pour des raisons de sécurité (pas d'exposition sur le WAN).



Configuration de l'Agent Zabbix

Pour bénéficier des fonctionnalités avancées de Zabbix (découverte automatique précise, métriques système détaillées), j'ai installé et configuré le plugin os-zabbix-agent.

- **Activation :** Service activé au démarrage.
- **Serveurs Zabbix :** J'ai restreint l'accès à l'adresse IP de notre serveur de supervision (192.168.1.30) pour sécuriser l'agent. Seul ce serveur peut envoyer des requêtes.
- **Port d'écoute :** Le port standard 10050 est utilisé.
- **Privilèges :** L'option Enable sudo root permissions a été cochée pour permettre à l'agent d'accéder à certaines métriques système protégées.



Configuration des règles de filtrage (Firewall)

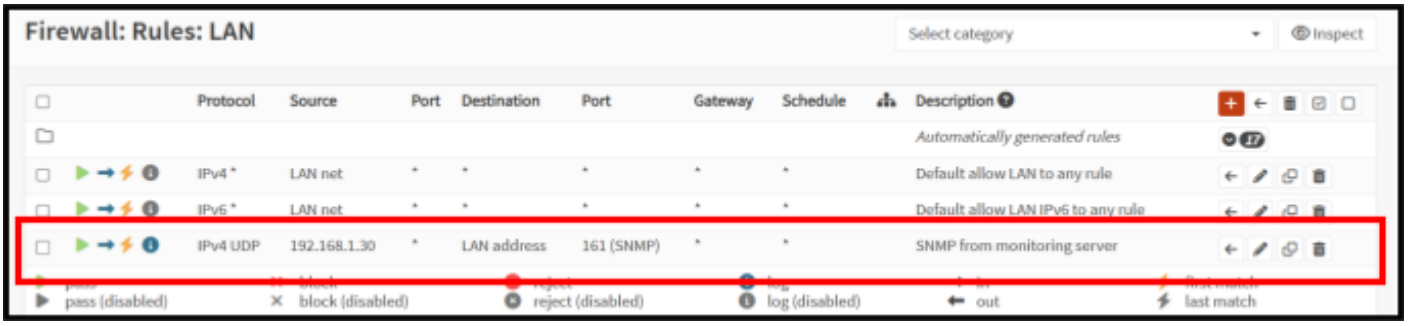
Par défaut, OPNsense bloque tout trafic entrant qui n'est pas explicitement autorisé. Pour que le serveur de supervision puisse interroger l'agent Zabbix et le service SNMP, j'ai dû créer des règles de flux spécifiques sur l'interface LAN.

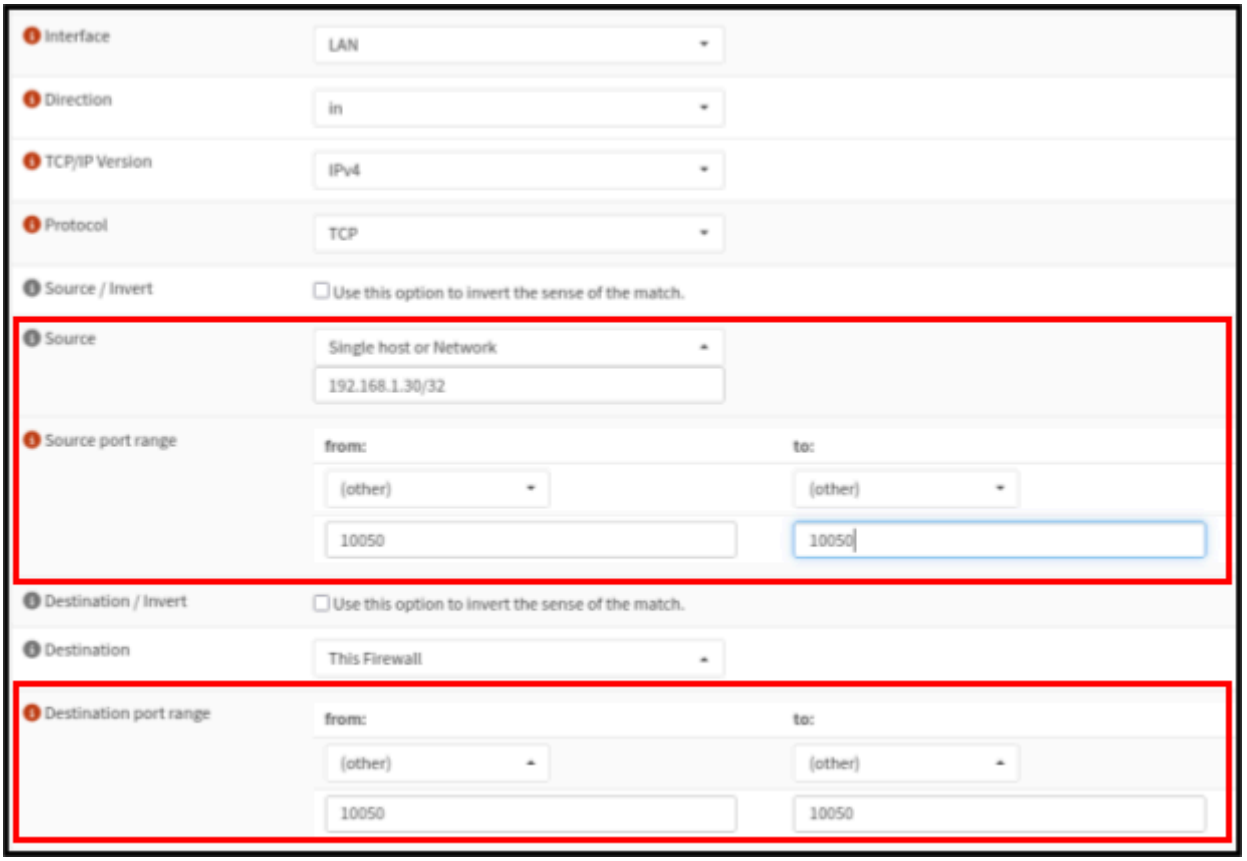
Règles mises en place :

1. **Autorisation SNMP :** Ouverture du flux UDP sur le port 161 depuis le serveur de supervision (192.168.1.30) vers le pare-feu.
2. **Autorisation Agent Zabbix :** Ouverture du flux TCP sur le port 10050.

Détail de la configuration de la règle Zabbix :

- **Interface :** LAN
- **Protocole :** TCP
- **Source :** 192.168.1.30/32 (L'IP unique du serveur de supervision, pour limiter la surface d'attaque).
- **Destination :** This Firewall (Le pare-feu lui-même).
- **Port de destination :** 10050 (Port de l'agent Zabbix).





Installation du serveur Zabbix

Nous avons installé le serveur Zabbix sur **192.168.1.30 (Debian 12)** en suivant ces étapes :

1. **Mettre à jour le système :**

```
sudo apt update && sudo apt upgrade -y
```

2. **Installer le dépôt Zabbix :**

```
wget https://repo.zabbix.com/zabbix/6.4/debian/pool/main/z/zabbix-release/zabbix-release_6.4-1+debi
```

```
an12_all.deb
sudo dpkg -i zabbix-release_6.4-1+debian12_all.deb
sudo apt update
```

3. Installer Zabbix Server, Frontend et Agent :

```
sudo apt install -y zabbix-server-mysql zabbix-frontend-php zabbix-apache-conf
zabbix-sql-scripts zabbix-agent
```

4. Installer la base de données MySQL pour Zabbix :

```
sudo mysql -u root -p
CREATE DATABASE zabbix CHARACTER SET UTF8MB4 COLLATE UTF8MB4_BIN;
CREATE USER 'zabbix'@'localhost' IDENTIFIED BY 'zabbixpass';
GRANT ALL PRIVILEGES ON zabbix.* TO 'zabbix'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

5. Importer le schéma de la base :

```
zcat /usr/share/doc/zabbix-sql-scripts/mysql/create.sql.gz | mysql -uzabbix -pzabbixpass zabbix
```

6. Configurer le fichier /etc/zabbix/zabbix_server.conf :

```
DBName=zabbix
DBUser=zabbix
DBPassword=zabbixpass
```

7. Démarrer et activer les services :

```
sudo systemctl restart zabbix-server zabbix-agent apache2
sudo systemctl enable zabbix-server zabbix-agent apache2
```

8. Accéder au frontend Zabbix via navigateur :

```
http://192.168.1.30/zabbix
```

4. Configuration des agents

1. **Agent Zabbix sur Linux (srv-linux 192.168.1.65) :**

```
sudo apt install zabbix-agent -y
sudo nano /etc/zabbix/zabbix_agentd.conf
Server=192.168.1.30
ServerActive=192.168.1.30
Hostname=srv-linux
sudo systemctl restart zabbix-agent
```

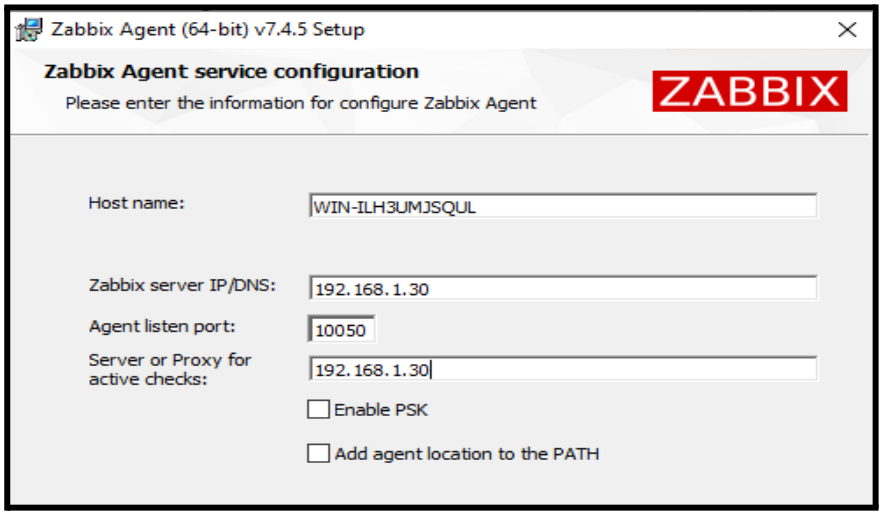
2. Agent Zabbix sur Windows (srv-windows 192.168.1.66)

- Configuration dans le fichier zabbix_agentd.conf :

Configuration de l'installation de zabbix :

Lors de l'installation de l'agent, j'ai défini les paramètres de connexion pour permettre la communication avec notre serveur de supervision central (srv-monitoring).

- **Hostname :** WIN-ILH3UMJSQUL (Ce nom d'hôte devra être identique lors de la déclaration de la machine dans l'interface web de Zabbix).
- **Zabbix Server IP :** 192.168.1.30 (L'agent n'acceptera les requêtes venant **que** de cette adresse IP pour des raisons de sécurité).
- **Agent listen port :** 10050 (Port standard d'écoute de l'agent).
- **Server or Proxy for active checks :** 192.168.1.30 (Permet à l'agent d'envoyer lui-même des données au serveur, utile pour les logs).



Configuration du Pare-feu Windows et Validation

Par défaut, le pare-feu Windows bloque les connexions entrantes. Pour autoriser le serveur de supervision à interroger l'agent, j'ai utilisé **PowerShell** pour créer une règle de flux spécifique plutôt que de passer par l'interface graphique, ce qui permet une configuration plus rapide et reproductible.

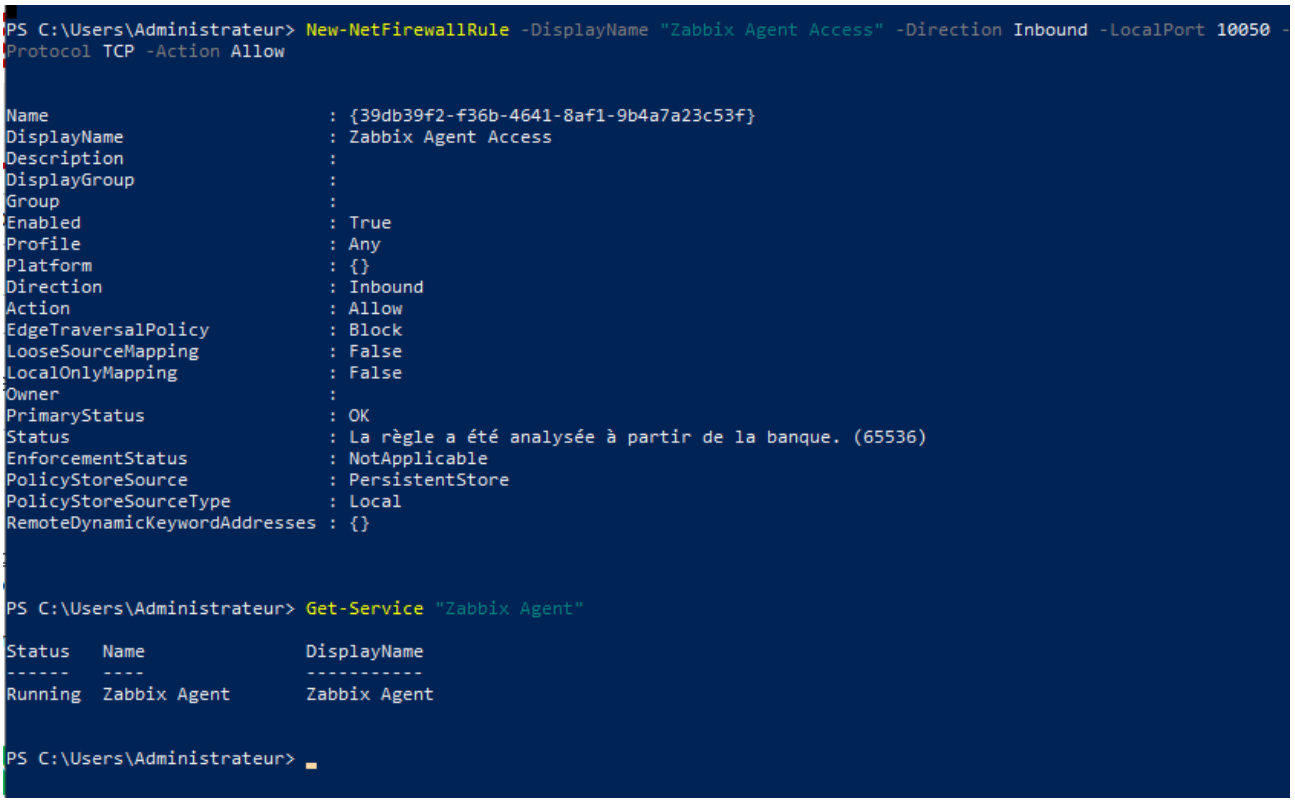
Actions réalisées :

1. **Ouverture de flux :** Utilisation de la commande New-NetFirewallRule pour autoriser le trafic entrant (Inbound) sur le port TCP 10050.
2. **Vérification du service :** Utilisation de la commande Get-Service pour confirmer que le service "Zabbix Agent" est bien en cours d'exécution (*Running*).

Commandes utilisées :

PowerShell
Création de la règle pare-feu
New-NetFirewallRule -DisplayName "Zabbix Agent Access" -Direction Inbound -LocalPort 10050 -Protocol TCP -Action Allow

Vérification de l'état du service
Get-Service "Zabbix Agent"



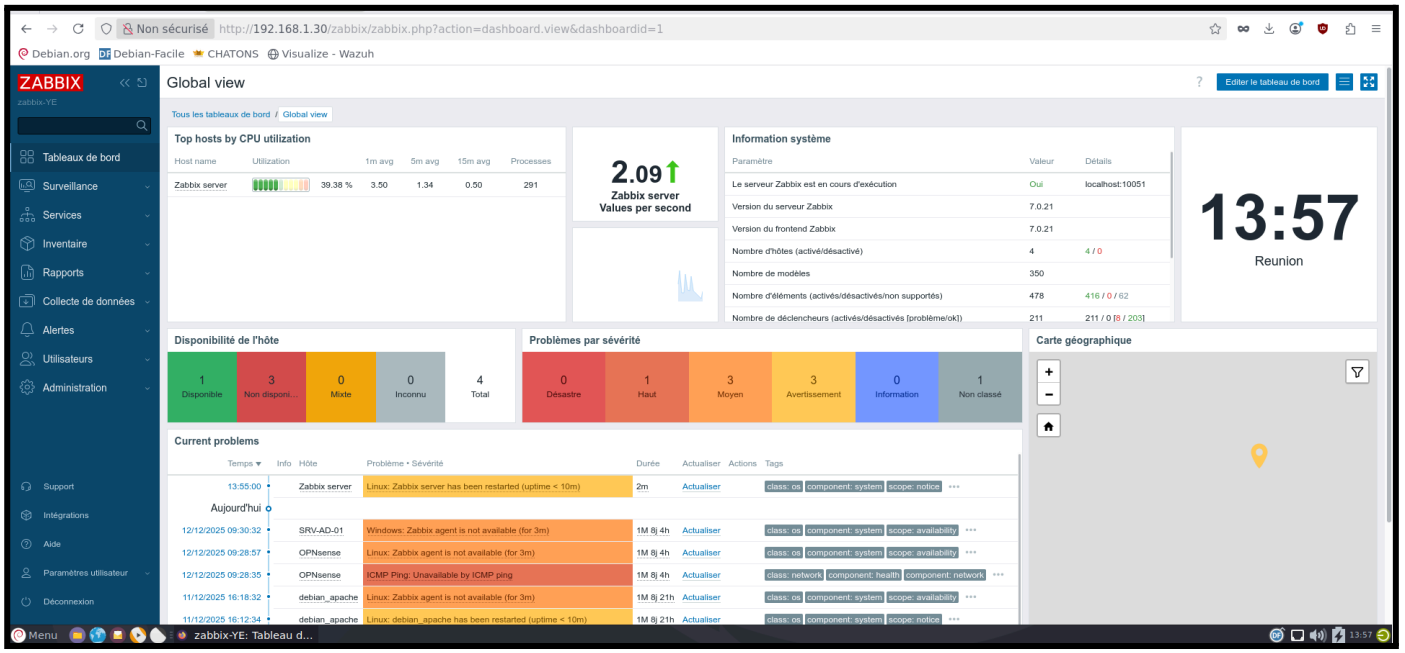
Démarrage du service Zabbix Agent.

3.4 Ajout des hôtes dans Zabbix

Dans le frontend Zabbix, nous avons ajouté :

- Linux : srv-linux
- Windows : srv-windows
- Pare-feu SNMP : OPNSENSE

Capture du dashboard Zabbix



5 Installation et supervision des hôtes

Les hôtes suivants ont été ajoutés dans Zabbix :

- Serveur Apache
- Serveur GLPI
- Serveur Monitoring
- OPNsense
- Serveur AD

Capture du tableau de bord Zabbix avec les hôtes supervisés

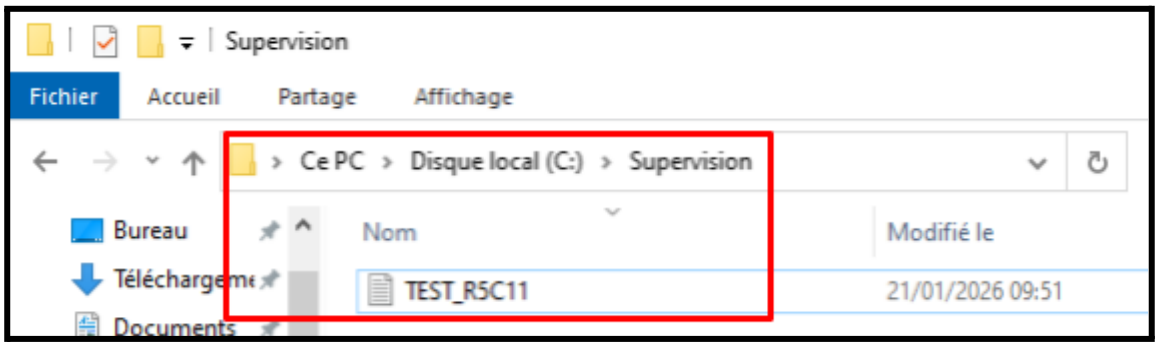
Nom ▲	Éléments	Déclencheurs	Graphiques	Découverte	Web	Interface
debian_apache	Éléments 92	Déclencheurs 28	Graphiques 17	Découverte 5	Web	192.168.1.65:10050
OPNsense	Éléments 157	Déclencheurs 71	Graphiques 35	Découverte 3	Web	192.168.1.1:10050
SRV-AD-01	Éléments 83	Déclencheurs 34	Graphiques 15	Découverte 4	Web	192.168.1.100:10050
Zabbix server	Éléments 146	Déclencheurs 78	Graphiques 14	Découverte 6	Web	127.0.0.1:10050

Transition vers la sécurité : Alors que Zabbix nous permet désormais de savoir *si* un serveur est allumé et performant, il ne nous permet pas d'analyser en profondeur *ce qui s'y passe* (qui se connecte, quel fichier est modifié, quelle attaque est tentée). Pour combler ce besoin de sécurité, nous déployons une solution de SIEM (Security Information and Event Management).

Création du fichier témoin

Dans un premier temps, j'ai créé un répertoire et un fichier spécifique sur le serveur Windows cible.

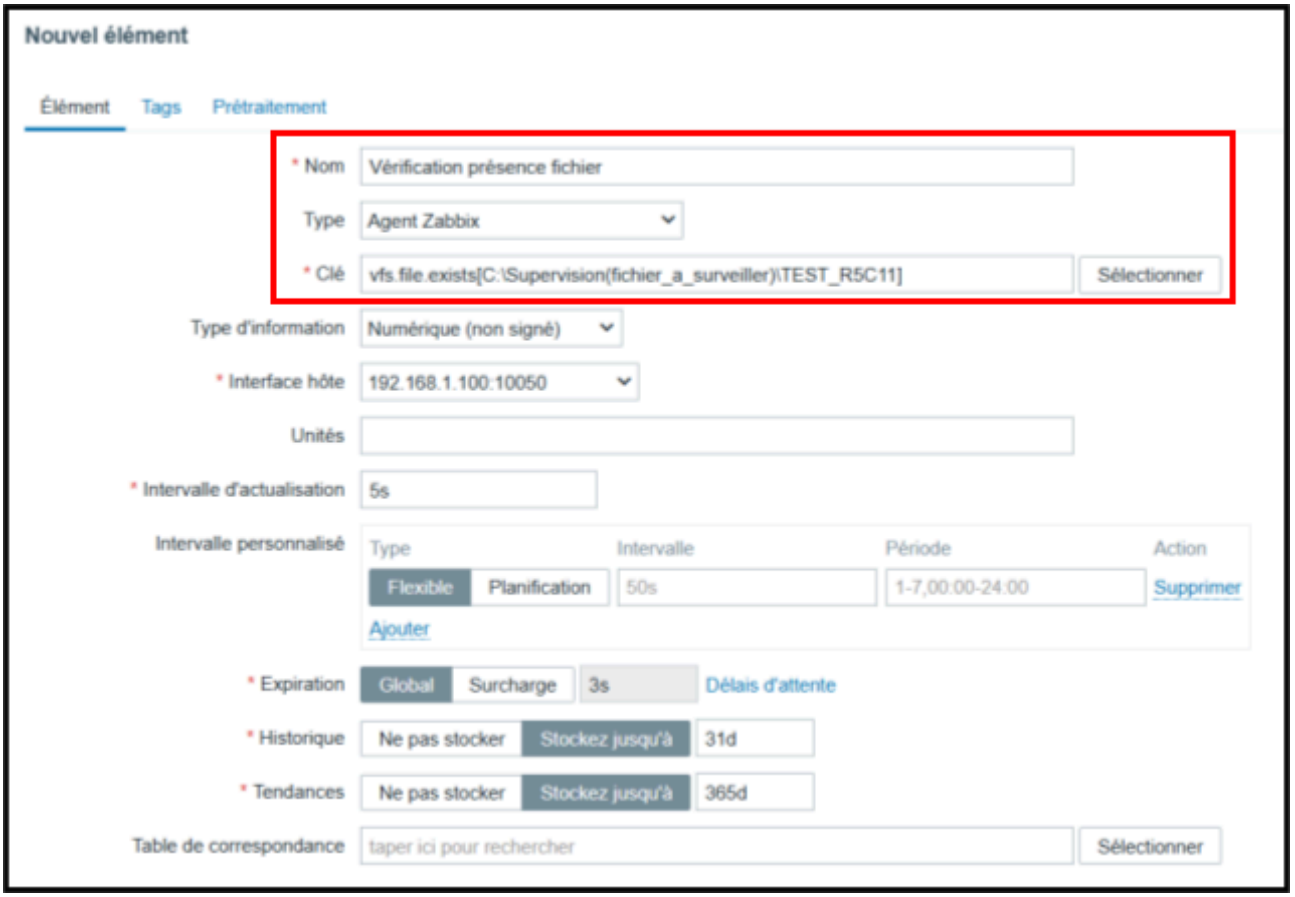
- **Chemin :** C:\Supervision
- **Fichier :** TEST_R5C11



Configuration de l'Élément (Item) dans Zabbix

J'ai configuré un nouvel **élément** associé à l'hôte SRV-AD-01. Cet élément utilise l'agent Zabbix pour interroger le système de fichiers.

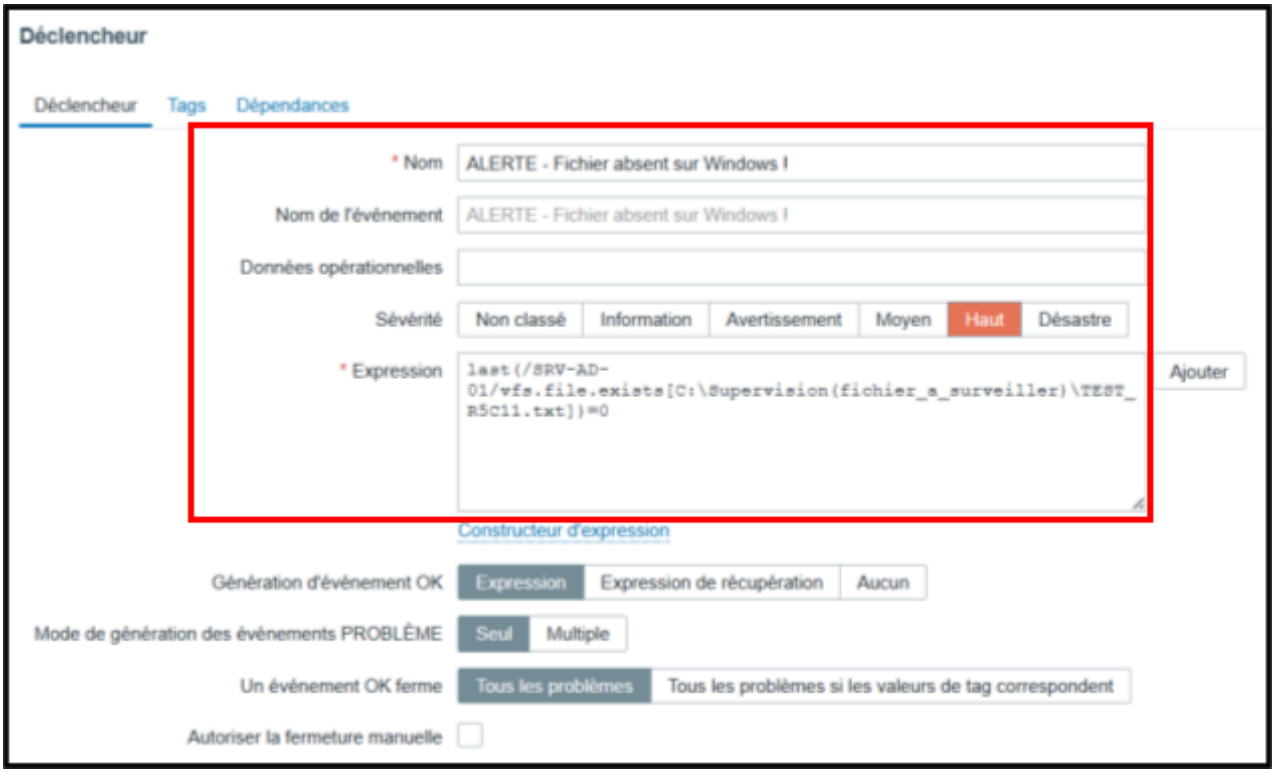
- **Clé utilisée :** vfs.file.exists[Chemin_du_fichier]
- **Fonctionnement :** Cette clé renvoie la valeur 1 si le fichier est présent, et 0 s'il est absent.
- **Intervalle :** La vérification est effectuée toutes les 5 secondes pour le test (intervalle réduit pour les besoins de la démonstration).



3.3 Configuration du Déclencheur (Trigger)

Une fois la donnée collectée, j'ai créé un **déclencheur** pour interpréter cette donnée.

- **Expression :** last(/SRV-AD-01/vfs.file.exists[...])=0
- **Logique :** Si la dernière valeur remontée par l'agent est égale à 0 (Faux), alors Zabbix active l'alerte.
- **Sévérité :** Classée comme "Haut", car la disparition d'un fichier de production peut être critique.

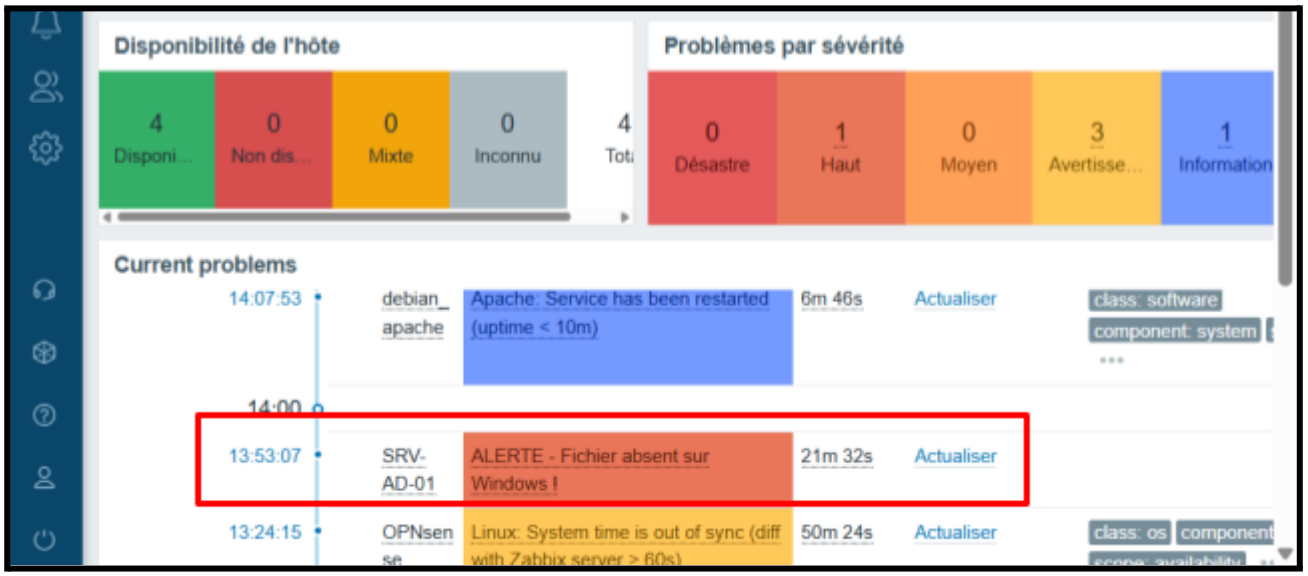


Simulation de l'incident et Résultat

Pour tester le bon fonctionnement de la chaîne d'alerte, j'ai procédé à la modification du nom du fichier sur le serveur Windows (le rendant ainsi "introuvable" pour l'agent Zabbix).

Résultat constaté :

1. L'agent Zabbix a détecté que le fichier TEST_R5C11 ne correspondait plus au chemin attendu.
2. La valeur 0 a été remontée au serveur.
3. Le déclencheur s'est activé quasi instantanément.
4. L'alerte **"ALERTE - Fichier absent sur Windows !"** est apparue dans le tableau de bord global, confirmant la détection de l'incident.



6. SOC / SIEM avec Wazuh

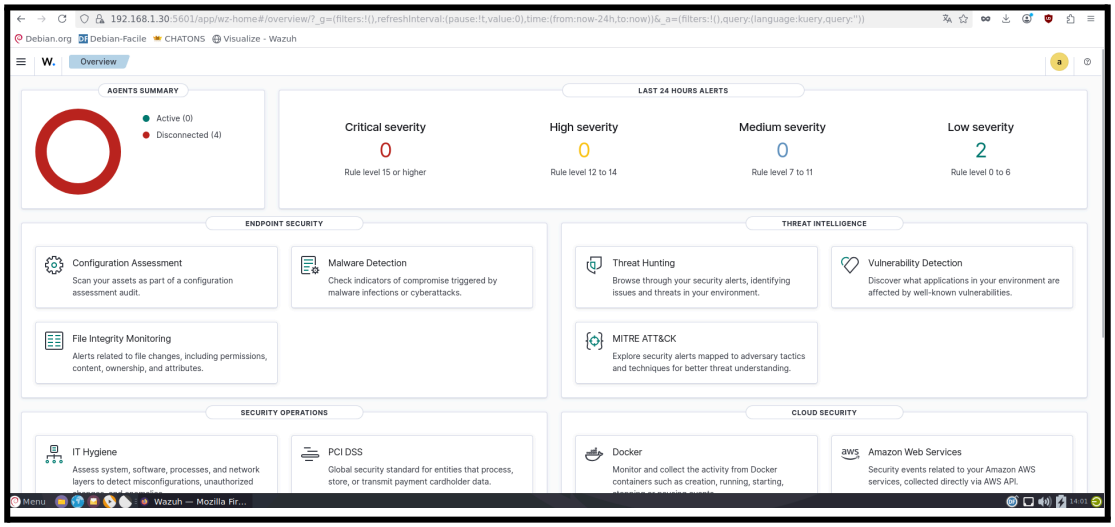
6.1 Présentation

Wazuh est utilisé comme **SIEM** afin de :

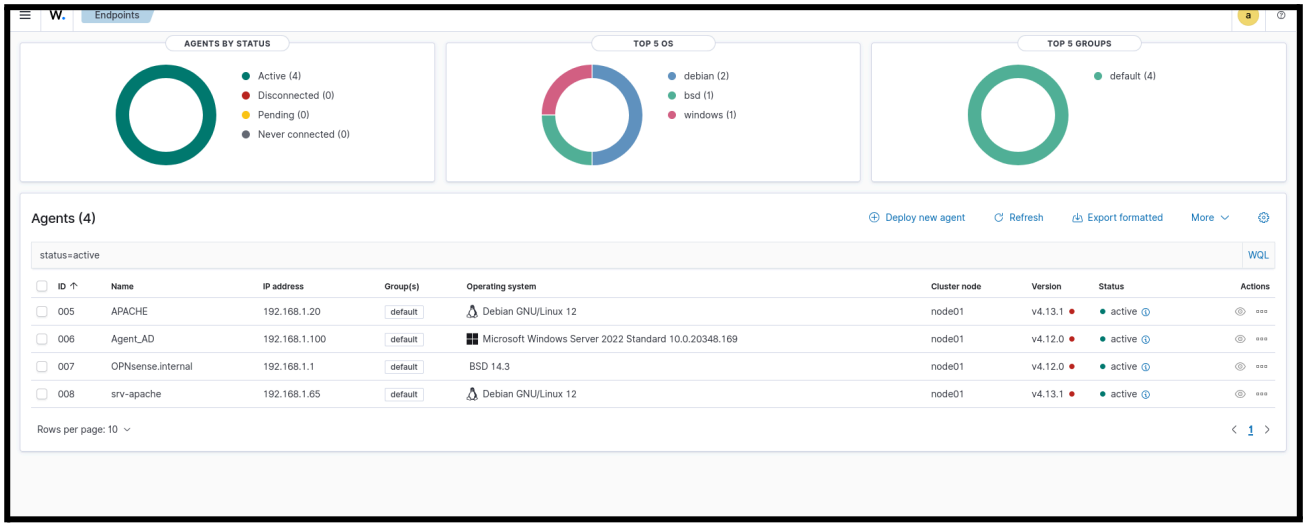
- centraliser les logs,
- détecter les incidents de sécurité,
- corréler les événements,
- déclencher des actions automatiques.

Interface Wazuh :
https://192.168.1.30:5601

Capture de l'interface Wazuh



Tous les hôtes actives sur Wazuh



6.2 Déploiement des agents

Un agent Wazuh a été installé sur :

- le serveur Apache (192.168.1.65),
- le serveur GLPI,
- le serveur Monitoring.

Un conflit de nom d'agent est apparu (nom par défaut *debian*).

Correction appliquée sur le serveur Apache :

```
<client>
  <enrollment>
    <agent_name>srv-apache</agent_name>
  </enrollment>
</client>
```

Capture de la commande agent_control -l montrant l'agent actif

Ce screen ci dessous montre tous les agents actif

```
root@debian:/home/etudiant# /var/ossec/bin/agent_control -l

Wazuh agent_control. List of available agents:
   ID: 000, Name: debian (server), IP: 127.0.0.1, Active/Local
   ID: 005, Name: APACHE, IP: 192.168.1.20, Disconnected
   ID: 006, Name: Agent_AD, IP: any, Disconnected
   ID: 007, Name: OPNsense.internal, IP: any, Disconnected
   ID: 008, Name: srv-apache, IP: any, Disconnected

List of agentless devices:
```

7. Mise en place de l'ITSM avec GLPI

7.1 Installation de GLPI (LAMP)

GLPI a été installé sur un serveur dédié.

Installation des prérequis :

```
apt update
apt install -y apache2 mariadb-server php php-xml php-common php-json php-mysql
php-mbstring php-curl php-gd php-intl php-zip php-bz2 php-imap php-apcu libapache2-mod-php
```

7.2 Configuration de la base de données

```
CREATE DATABASE glpidb;
CREATE USER 'glpi'@'localhost' IDENTIFIED BY 'yann';
GRANT ALL PRIVILEGES ON glpidb.* TO 'glpi'@'localhost';
FLUSH PRIVILEGES;
```

7.3 Installation de GLPI

GLPI version 10 a été installée dans **/var/www/html/glpi.**

Accès web :
http://192.168.1.20/glpi

Identifiants initiaux :

- Login : glpi

- Mot de passe : glpi
  Mot de passe changé en : **yann123**

Capture de la page du tableau de bord GLPI

8. Interconnexion Wazuh → GLPI (API REST)

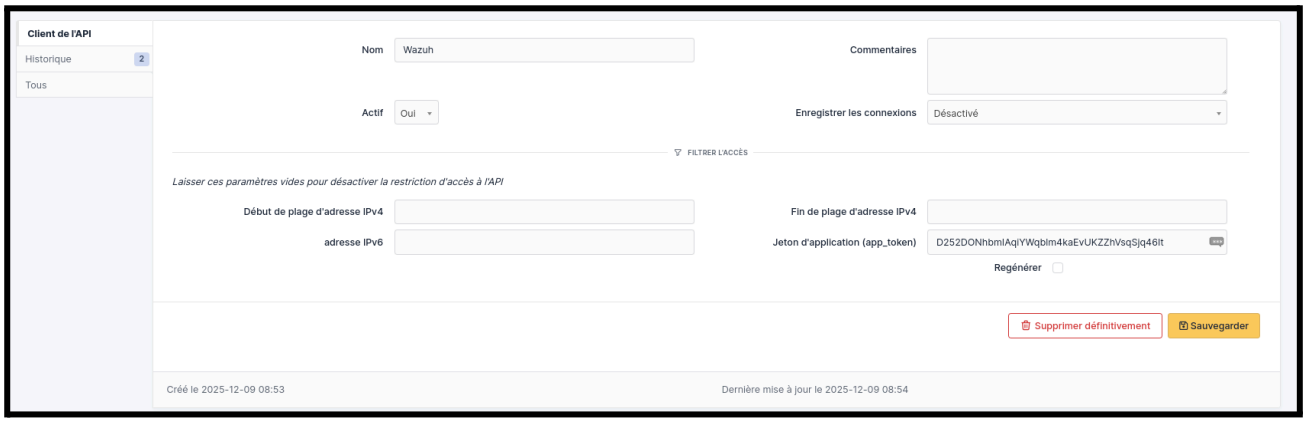
8.1 Activation de l'API GLPI

API activée dans **Configuration > Générale > API**.

Jetons utilisés :(Ce sont de faux token juste pour le TP pour montrer à quoi ressemble les token)

- App Token : D252DONhbmIAqiYWqblm4kaEvUKZZhVsqSjq46It
- User Token : mE022JUlo1RZ0ArkmHaKw7u7LXssC0fHj2C368Z7

Capture de la configuration API



8.2 Script d'intégration personnalisé

Un script Python a été développé sur le serveur Wazuh afin de :

- lire une alerte Wazuh (JSON),
- déterminer la priorité,
- créer automatiquement un ticket GLPI via l'API REST.

Script Python complet:

```
#!/var/ossec/framework/python/bin/python3
import sys
import json
import requests

# --- CONFIGURATION ---
glpi_ip = "192.168.1.20"
app_token = "D252DONhbmIAqiYWqblm4kaEvUKZZhVsqSjq46It"
user_token = "mE022JUlo1RZ0ArkmHaKw7u7LXssC0fHj2C368Z7"

# --- LOGIQUE DE PRIORITÉ ---
def get_priority(level):
    if level >= 12:
        return 5, "[CRITIQUE]"  # P1
    elif level >= 10:
        return 4, "[ALERTE]"    # P2
    elif level >= 7:
        return 3, "[ATTENTION]" # P3
    else:
        return 2, "[INFO]"      # P4

# --- DÉBUT DU TRAITEMENT ---
session = requests.Session()
session.trust_env = False

# Lire l'alerte Wazuh
try:
    with open(sys.argv[1]) as f:
        alert_json = json.load(f)
except:
    sys.exit()

# Niveau d'alerte
wazuh_level = int(alert_json['rule']['level'])
glpi_priority, prefix = get_priority(wazuh_level)

# Connexion GLPI
url_init = f"http://{glpi_ip}/glpi/apirest.php/initSession"
headers_init = {"App-Token": app_token, "Authorization": f"user_token {user_token}"}
```

```
try:
    response = session.get(url_init, headers=headers_init, verify=False)
    session_token = response.json().get('session_token')
    if not session_token: sys.exit()
except:
    sys.exit()

# Création du ticket
description = alert_json['rule']['description']
agent_name = alert_json['agent']['name']
full_log = alert_json.get('full_log', 'Pas de log brut')

ticket_data = {
    "input": {
        "name": f"{prefix} {description}",
        "content": f"Agent: {agent_name}\nNiveau Wazuh: {wazuh_level}\n\nDétails:\n{full_log}",
        "priority": glpi_priority,
        "type": 1
    }
}

url_ticket = f"http://{glpi_ip}/glpi/apirest.php/Ticket/"
headers_ticket = {"App-Token": app_token, "Session-Token": session_token, "Content-Type":
"application/json"}

try:
    session.post(url_ticket, headers=headers_ticket, json=ticket_data, verify=False)
except:
    sys.exit()
```

La priorité est définie selon le niveau Wazuh :

- Level ≥ 12 → P1 Critique
- Level ≥ 10 → P2 Haute (Brute Force)
- Level ≥ 7 → P3
- Sinon → P4

Ce script permet de transformer automatiquement une alerte Wazuh en ticket GLPI avec la priorité correspondant au niveau de l'alerte.

9. Configuration Wazuh – Déclaration du script

Dans Wazuh, pour qu'un **script externe soit exécuté automatiquement** lorsqu'une alerte est détectée, il faut déclarer ce script dans le fichier de configuration principal du manager Wazuh :

```
/var/ossec/etc/ossec.conf
```

Nous avons ajouté la section suivante :

```
<integration>
  <name>custom-glpi</name>
  <hook_url>http://192.168.1.20/glpi/apirest.php</hook_url>
  <level>10</level>
  <alert_format>json</alert_format>
</integration>
```

Explications des balises

<name>

Nom de l'intégration Wazuh.
custom-glpi identifie le script et apparaît dans les logs.

<hook_url>

URL de destination des alertes.
Ici : API REST GLPI (http://192.168.1.20/glpi/apirest.php).

<level>

Niveau minimum d'alerte déclenchant l'intégration.
Valeur 10 : seules les alertes critiques sont envoyées.

<alert_format>

Format des données envoyées.
json, facilement exploitable par le script Python.

Pourquoi faire cela ?

- Pour que Wazuh **puisse déclencher automatiquement une action externe** lorsqu'une alerte critique survient.
- Ici, l'action est : **créer un ticket dans GLPI** sans intervention humaine.
- Cela respecte la **logique ITIL / ITSM** :

- Détection → Notification → Ticket → Priorisation → Suivi

- Évite le **retard humain** : un administrateur n'a pas besoin de surveiller constamment Wazuh.

- Permet de gérer les incidents de manière **automatisée et centralisée**, ce qui est exactement l'objectif de ce TP.

10. Automatisation de la gestion des incidents - Création d'un ticket critique (P1) dans GLPI suite à la perte de connexion d'un serveur.

Cette capture d'écran valide l'intégration complète entre notre SIEM (Wazuh) et notre outil ITSM (GLPI). Elle démontre les trois points attendus :

1. **Mise en place de l'ITSM :** L'interface montre l'outil **GLPI** opérationnel et configuré pour la gestion des tickets.
2. **Remontée Automatique des Alertes :**

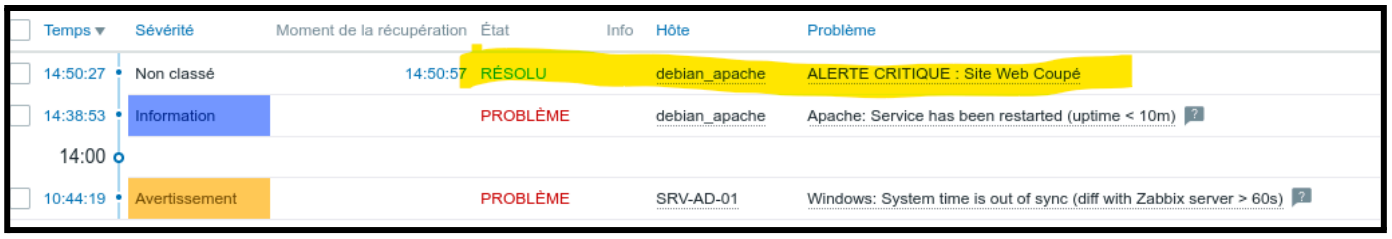
- Le ticket a été généré automatiquement suite à la détection d'un incident technique.
- Le titre du ticket **"[CRITIQUE]... Hôte éteint"** et la description (indiquant l'agent srv-apache et l'alerte Keepalive expired) prouvent que les données techniques ont été correctement transmises du superviseur vers le système de ticket sans intervention humaine.

3. **Gestion de la Priorisation (Criticité) :**

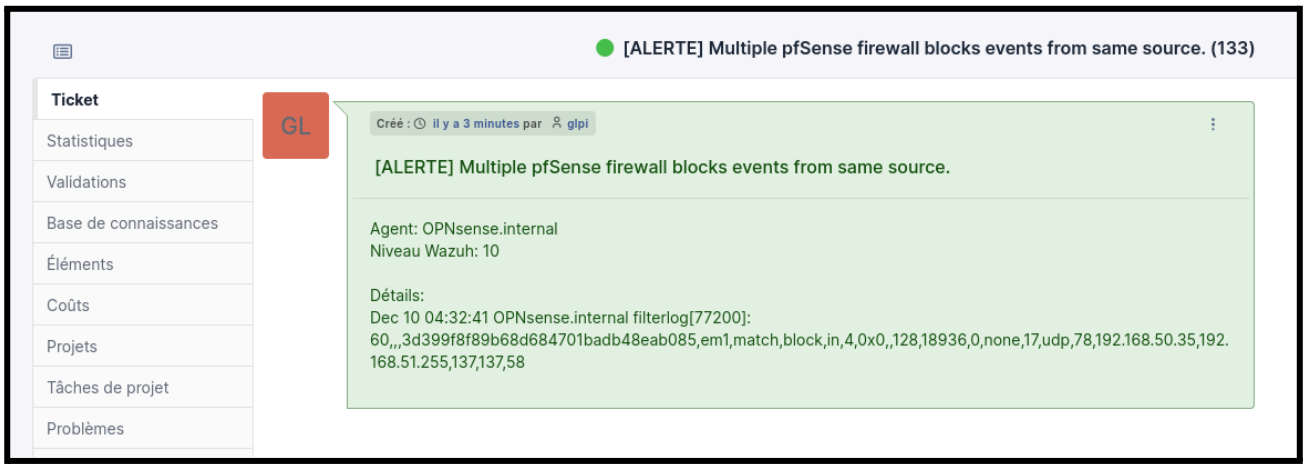
- Comme demandé, une règle de correspondance a été appliquée.
- L'incident étant une perte totale de communication avec un serveur (Hôte éteint), le script a automatiquement classé ce ticket en **Priorité : Très haute** (visible dans la colonne de droite).
- Cela correspond au niveau de criticité **P1** attendu pour un arrêt de service, déclenchant une prise en charge immédiate.

Supervision Zabbix

On voit ci dessous que l'alerte "Service HTTP Down" et afficher aussi sur Zabbix



Intégration SIEM/ITSM - Création automatique d'un ticket d'incident de sécurité suite à une détection de flux suspects sur le pare-feu:



Cette capture valide la capacité de l'infrastructure à traiter des incidents de sécurité en temps réel via l'outil de ticketing GLPI. Elle répond aux exigences du projet sur les points suivants :

1. **Centralisation des incidents (ITSM) :**
2. **Scénario de Sécurité (Remontée d'alertes) :**
3. **Gestion de la Priorité (Contextuelle) :**

Intégration SIEM/ITSM - Création automatique d'un ticket d'incident de sécurité (Brute Force SSH):

Cette capture valide la capacité de l'infrastructure à traiter des incidents de sécurité en temps réel via l'outil de ticketing GLPI. Elle répond aux exigences du projet sur les points suivants :

1. **Centralisation des incidents (ITSM) :** Le ticket a été créé automatiquement via l'API REST sans intervention humaine, suite à une alerte remontée par le SIEM.
2. **Scénario de Sécurité (Remontée d'alertes) :** L'incident concerne une tentative d'intrusion par force brute sur le service SSH (sshd: brute force...). Les détails techniques (IP source,

utilisateur invalide, heure) sont directement inclus dans la description du ticket, facilitant l'analyse pour l'administrateur.

3. **Gestion de la Priorité (Contextuelle) :** On constate que le ticket a été classé avec une **Priorité : Haute**. Cela confirme le bon fonctionnement du script Python qui a traduit le "Niveau Wazuh : 10" (visible dans le ticket) en une priorité élevée dans GLPI

Afin de valider la réactivité de notre infrastructure de supervision, j'ai mis en place un scénario de test consistant à surveiller la présence d'un fichier critique sur le serveur Windows (SRV-AD-01) et à déclencher une alerte si ce fichier disparaît ou est renommé.

11-Analyse du fichier de configuration principal (ossec.conf)

Pour centraliser la sécurité de l'infrastructure et répondre aux exigences du TP, le fichier de configuration du Manager Wazuh a été paramétré comme suit. Ce fichier dicte la manière dont le SOC analyse les logs et réagit aux menaces.

Fichier Ossec.conf ci-dessous:

```
<ossec_config>
  <global>
    <jsonout_output>yes</jsonout_output>
    <alerts_log>yes</alerts_log>
    <logall>no</logall>
    <agents_disconnection_time>10m</agents_disconnection_time>
    <white_list>127.0.0.1</white_list>
    <white_list>192.168.1.20</white_list> </global>

  <integration>
```

```
    <name>custom-glpi</name>
    <hook_url>http://192.168.1.20/glpi/apirest.php/Ticket/</hook_url>
    <level>7</level>
  <group>authentication_failed,brute_force,rootkit,vulnerability_detector</group>
    <alert_format>json</alert_format>
  </integration>

  <syscheck>
    <disabled>no</disabled>
    <frequency>43200</frequency> <scan_on_start>yes</scan_on_start>
    <alert_new_files>yes</alert_new_files>

    <directories realtime="yes" check_all="yes">/etc,/usr/bin,/usr/sbin,/bin,/sbin</directories>

    <ignore type="sregex">.log$|.swp$|.tmp$</ignore>
  </syscheck>

  <vulnerability-detection>
    <enabled>yes</enabled>
    <index-status>yes</index-status>
    <feed-update-interval>60m</feed-update-interval>
  </vulnerability-detection>

  <active-response>
    <command>firewall-drop</command>
    <location>local</location>
    <rules_id>5712,5710</rules_id> <timeout>1800</timeout> </active-response>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/auth.log</location> </localfile>

  <localfile>
    <log_format>journald</log_format>
    <location>journald</location>
  </localfile>

</ossec_config>
```

A. Gestion des agents et connectivité

Dans la section <global>, j'ai défini les paramètres de base pour la communication avec les serveurs srv-linux et srv-windows :

- **jsonout_output :** Activé pour permettre l'export des alertes vers le tableau de bord.
- **agents_disconnection_time :** Configuré sur **10m**. Si un agent ne répond plus pendant ce délai, une alerte est générée.

- **white_list :** J'ai ajouté l'IP **192.168.1.20** (serveur GLPI/Monitoring) afin d'éviter que nos outils d'administration ne soient bloqués par une règle de sécurité automatique.

B. Interconnexion avec l'outil de Ticketing (Étape 5)

L'objectif est de transformer une alerte de sécurité en un ticket d'incident dans **GLPI**.

- **hook_url :** Pointe vers l'API REST de mon serveur GLPI.
- **level :** J'ai fixé le seuil au **niveau 7**. Cela garantit que seuls les incidents sérieux (tentatives d'intrusion, détection de malware) créent un ticket, évitant ainsi de polluer l'outil de gestion.
- **group :** Les alertes ciblées sont le brute-force, les rootkits et les vulnérabilités critiques.

C. Surveillance en temps réel et Intégrité (FIM)

Le module <syscheck> est configuré pour surveiller les fichiers système :

- **realtime="yes" :** La surveillance des dossiers /etc, /bin et /sbin se fait en temps réel. Toute modification d'un binaire système déclenche une alerte immédiate.
- **Filtre d'exclusion :** Les fichiers temporaires et les logs (.log, .tmp) sont ignorés via des expressions régulières pour ne pas générer de faux positifs.

D. Réponse Active : Protection contre le Brute-Force (Étape 4)

C'est ici que le SOC devient proactif. La section <active-response> permet de bloquer un attaquant sans intervention humaine :

- **Command :** Utilisation de firewall-drop.
- **Déclencheur :** Si les règles **5712** ou **5710** (Brute-force SSH détecté sur srv-linux) sont activées, l'IP source est immédiatement bannie.
- **Timeout :** Le blocage dure **1800 secondes** (30 minutes), ce qui suffit à décourager la plupart des scripts d'attaque automatisés.

E. Collecte des sources de logs

Enfin, j'ai spécifié les sources locales à analyser pour nourrir le moteur de corrélation :

- **auth.log :** Pour suivre les tentatives de connexion sur l'OS Linux.
- **journald :** Pour récupérer les événements système globaux.

12-Analyse des flux et Investigation:

Pour valider la chaîne de traitement des logs, j'ai utilisé l'interface d'investigation **Discover**. Cette vue est l'outil principal du SOC pour effectuer du "Threat Hunting" (recherche de menaces) ou de l'analyse "Forensics" après un incident.

Ce que la capture d'écran montre :

L'analyse de cette interface permet de confirmer plusieurs points critiques de notre infrastructure de sécurité :

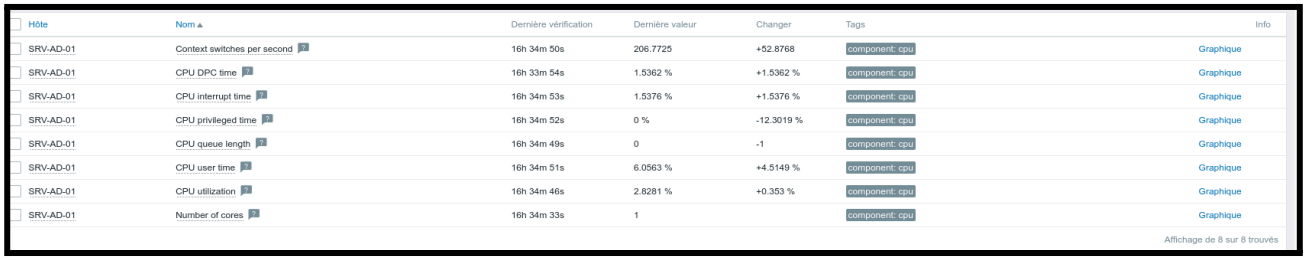
- Histogramme de charge : Le graphique temporel en haut de l'écran montre une activité constante de 785 événements. Cette visualisation permet de détecter immédiatement une anomalie : un pic soudain pourrait indiquer une attaque en cours, tandis qu'un creux signalerait la panne d'un agent ou d'un service.
- Filtrage multi-critères : J'ai appliqué des filtres spécifiques (visibles en haut à gauche) pour isoler les logs provenant de l'agent Agent_AD. Cela démontre la capacité du SIEM à trier des milliers de journaux pour se concentrer sur une machine précise lors d'un audit.
- Analyse d'un événement Kerberos (EventID 4769) : Le log détaillé sur la capture correspond à une requête de ticket de service Kerberos. On y voit des informations précieuses :
    - **Le Service :** krbtgt (le service de distribution de clés).
    - **L'Utilisateur :** Le compte machine WIN-ILH3UMJSQL$.
    - **L'Origine :** L'adresse IP source de la demande de ticket.

Interprétation technique : La remontée de l'EventID 4769 est cruciale. C'est l'un des événements les plus surveillés en cybersécurité car il permet de détecter des

techniques d'escalade de privilèges ou de mouvements latéraux au sein d'un domaine Active Directory.

13-Analyse de la performance et santé des hôtes (Focus CPU)

Au-delà de la simple vérification de disponibilité (Ping), la supervision avec Zabbix nous permet d'analyser en temps réel la charge des ressources critiques de nos serveurs. Pour le contrôleur de domaine (**SRV-AD-01**), nous avons configuré une surveillance approfondie de l'utilisation processeur.



Hôte	Nom ▲	Dernière vérification	Dernière valeur	Changer	Tags	Info
SRV-AD-01	Context switches per second	16h 34m 50s	206.7725	+52.8768	component: cpu	Graphique
SRV-AD-01	CPU DPC time	16h 33m 54s	1.5362 %	+1.5362 %	component: cpu	Graphique
SRV-AD-01	CPU interrupt time	16h 34m 53s	1.5376 %	+1.5376 %	component: cpu	Graphique
SRV-AD-01	CPU privileged time	16h 34m 52s	0 %	-12.3019 %	component: cpu	Graphique
SRV-AD-01	CPU queue length	16h 34m 49s	0	-1	component: cpu	Graphique
SRV-AD-01	CPU user time	16h 34m 51s	6.0563 %	+4.5149 %	component: cpu	Graphique
SRV-AD-01	CPU utilization	16h 34m 46s	2.8281 %	+0.353 %	component: cpu	Graphique
SRV-AD-01	Number of cores	16h 34m 33s	1		component: cpu	Graphique

Affichage de 8 sur 8 trouvés

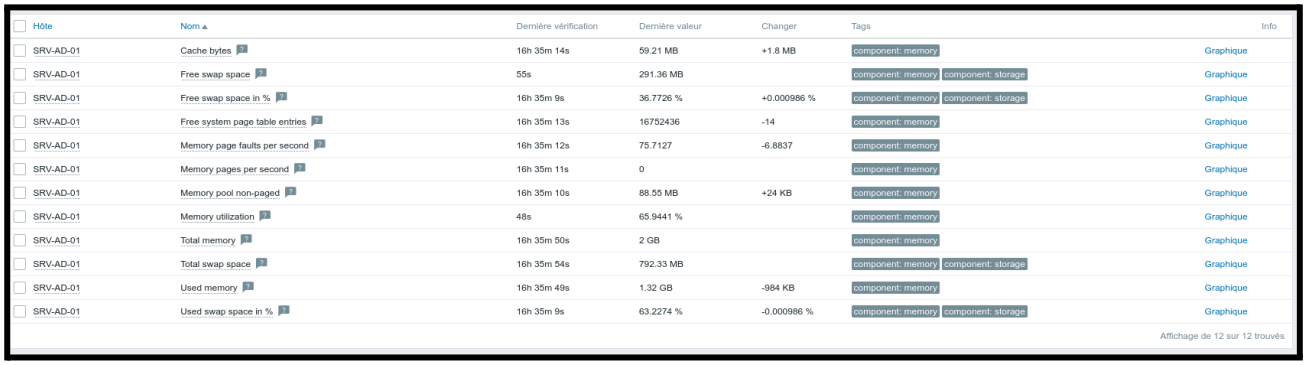
Interprétation des indicateurs :

- **Utilisation CPU (%) :** On observe que le serveur maintient une charge moyenne très basse (environ 2,8%). Cela confirme que les ressources allouées à la machine virtuelle sont suffisantes pour les services actuels.
- **CPU User vs System time :** Cette distinction permet de savoir si la charge est causée par des applications utilisateur ou par des processus système (noyau).
- **Context Switches :** Le suivi des commutations de contexte nous aide à identifier d'éventuels conflits de processus ou une surcharge logicielle qui pourrait ralentir l'authentification des utilisateurs sur le domaine.

Utilité pour le MCO : Cette visibilité est indispensable pour le **Maintien en Condition Opérationnelle**. En cas de ralentissement du réseau ou des services d'annuaire, ces graphiques permettent de lever le doute sur une éventuelle saturation matérielle avant de procéder à une investigation plus poussée dans les logs de sécurité.

Surveillance de la mémoire vive (RAM - SRV-AD-01)

La mémoire vive est une ressource critique pour un contrôleur de domaine, car elle stocke la base de données Active Directory en cache pour accélérer les authentifications. Nous surveillons l'utilisation de la RAM pour prévenir tout ralentissement des services de domaine.



Hôte	Nom ▲	Dernière vérification	Dernière valeur	Changer	Tags	Info
SRV-AD-01	Cache bytes	16h 35m 14s	59.21 MB	+1.8 MB	component: memory	Graphique
SRV-AD-01	Free swap space	55s	291.36 MB		component: memory component: storage	Graphique
SRV-AD-01	Free swap space in %	16h 35m 9s	36.7726 %	+0.000986 %	component: memory component: storage	Graphique
SRV-AD-01	Free system page table entries	16h 35m 13s	16752436	-14	component: memory	Graphique
SRV-AD-01	Memory page faults per second	16h 35m 12s	75.7127	-6.8837	component: memory	Graphique
SRV-AD-01	Memory pages per second	16h 35m 11s	0		component: memory	Graphique
SRV-AD-01	Memory pool non-paged	16h 35m 10s	88.55 MB	+24 KB	component: memory	Graphique
SRV-AD-01	Memory utilization	48s	65.9441 %		component: memory	Graphique
SRV-AD-01	Total memory	16h 35m 50s	2 GB		component: memory	Graphique
SRV-AD-01	Total swap space	16h 35m 54s	792.33 MB		component: memory component: storage	Graphique
SRV-AD-01	Used memory	16h 35m 49s	1.32 GB	-984 KB	component: memory	Graphique
SRV-AD-01	Used swap space in %	16h 35m 9s	63.2274 %	-0.000986 %	component: memory component: storage	Graphique

Affichage de 12 sur 12 trouvés

Analyse de la capture :

- **Memory utilization (%) :** Cet indicateur permet de s'assurer que le serveur dispose d'une marge de manœuvre suffisante. Une utilisation supérieure à 80% de manière constante indiquerait un besoin d'augmenter les ressources de la machine virtuelle.
- **Available memory :** Contrairement à la mémoire totale, la mémoire "disponible" inclut le cache, ce qui donne la vision réelle de ce que le système peut encore allouer aux services.
- **Swap / Pagefile usage :** Nous surveillons également l'utilisation du fichier d'échange. Une montée de l'utilisation du Swap indiquerait une saturation de la RAM physique, ce qui dégraderait fortement les performances du serveur.

Synthèse Performance : L'association des mesures CPU et RAM nous permet de confirmer que le serveur **SRV-AD-01** est correctement dimensionné. En cas d'alerte de sécurité remontée par Wazuh (ex: attaque par déni de service), ces graphiques de performance nous permettent de mesurer immédiatement l'impact sur la disponibilité réelle des services d'annuaire.

-14 Interconnexion Zabbix 7.0 & GLPI 10 via API REST

Étape 1 : Configuration de l'environnement GLPI

Nous avons commencé par configurer l'API REST de GLPI pour qu'elle accepte les connexions externes de manière sécurisée.
Nous nous sommes rendus dans **Configuration > Générale > API** et avons activé l'accès en lecture/écriture.

Ensuite, nous avons créé un **App-Token** nommé **"Zabbix-Server"** pour que GLPI reconnaisse l'application autorisée à se connecter.
Nous avons également ajouté, en option, l'adresse IP du serveur Zabbix dans la liste blanche afin de renforcer la sécurité.

Pour sécuriser l'accès utilisateur, nous avons sélectionné le compte glpi dans Administration > Utilisateurs et généré un Personal Token (User-Token) sous l'onglet Accès distants.
Ce jeton nous a permis de nous authentifier via le script sans exposer le mot de passe en clair.

Étape 2 : Développement et installation du script "Bridge"

Nous avons ensuite développé un script Bash qui fait office de traducteur entre les alertes Zabbix et le format JSON exigé par GLPI.

Nous avons créé le fichier **/usr/lib/zabbix/alertscripts/zabbix_to_glpi.sh** et y avons inséré le script suivant :

```
#!/bin/bash

# --- CONFIGURATION ---
GLPI_URL="http://192.168.1.20/glpi/apirest.php"
APP_TOKEN="D252DONhbmIAqiYWqblm4kaEvUKZZhVsqSjq46It"
USER_TOKEN="mE022JUlo1RZ0ArkmHaKw7u7LXssC0fHj2C368Z7"

# --- TRAITEMENT DES DONNÉES ---
SUBJECT=$(echo "$2" | tr -d '"' | tr -d '\r' | tr '\n' ' ')
MESSAGE=$(echo "$3" | tr -d '"' | tr -d '\r' | tr '\n' ' ')

# 1. Initialisation de la session API
RESPONSE=$(curl -s -X GET "$GLPI_URL/initSession" \
    -H "App-Token: $APP_TOKEN" \
    -H "Authorization: user_token $USER_TOKEN")

SESSION_TOKEN=$(echo $RESPONSE | sed 's/.*session_token":"\([^"]*\)".*/\1/')

# 2. Création du ticket si l'authentification est réussie
if [[ $SESSION_TOKEN =~ ^[a-z0-9]+$ ]]; then
    curl -s -X POST "$GLPI_URL/Ticket/" \
        -H "App-Token: $APP_TOKEN" \
        -H "Session-Token: $SESSION_TOKEN" \
        -H "Content-Type: application/json" \
        -d "{
            \"input\": {
                \"name\": \"$SUBJECT\",
                \"content\": \"$MESSAGE\",
                \"entities_id\": 0,
                \"type\": 1
            }
        }"
fi
```

Nous avons ensuite ajusté les permissions pour que l'utilisateur système **zabbix** puisse exécuter ce script :

```
sudo chown zabbix:zabbix /usr/lib/zabbix/alertscripts/zabbix_to_glpi.sh
```

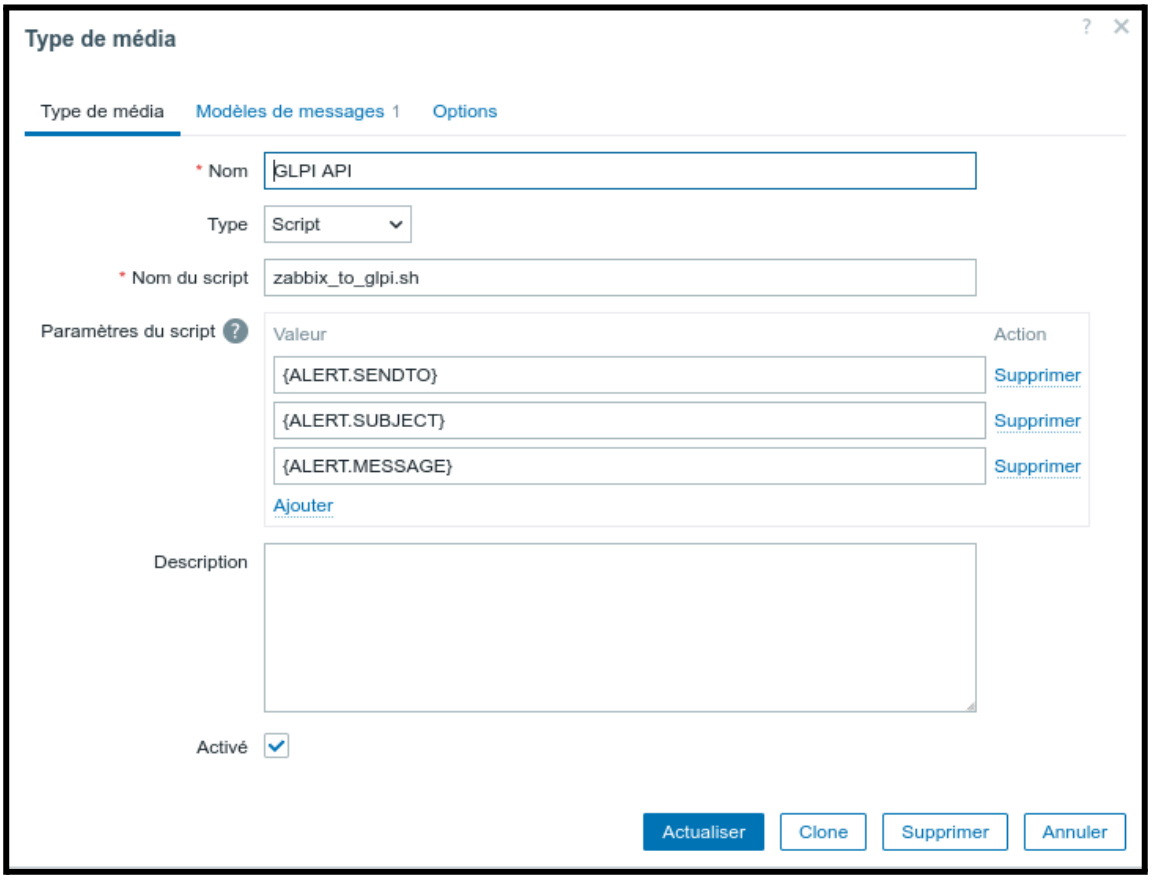
```
sudo chmod 755 /usr/lib/zabbix/alertscripts/zabbix_to_glpi.sh
```

Étape 3 : Paramétrage du flux d'alerte dans Zabbix

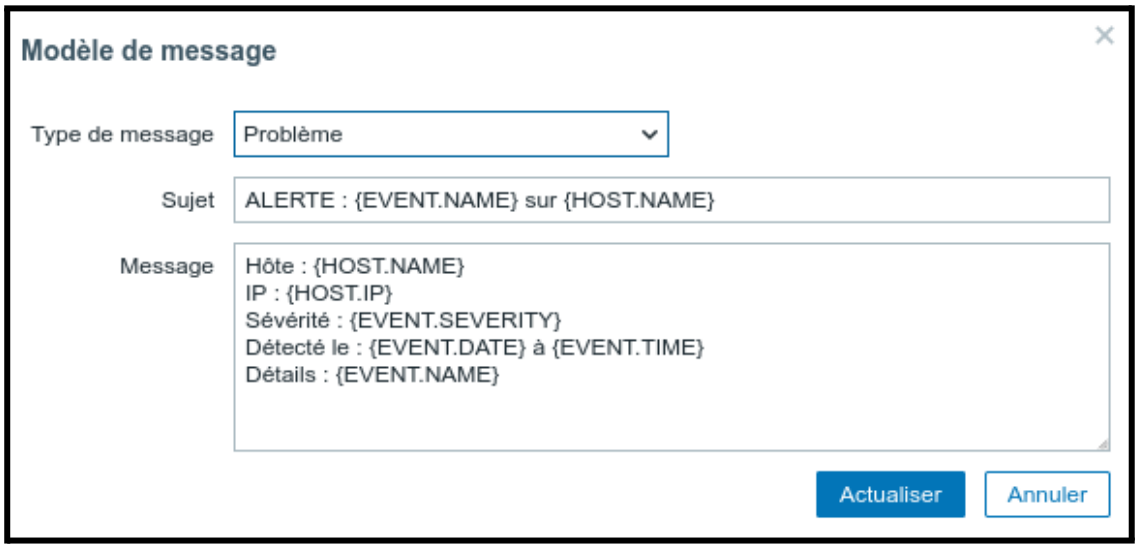
Nous avons poursuivi en configurant Zabbix pour déclencher notre script.
Nous avons créé un média nommé "GLPI API" dans **Alerts > Media types**, de type Script, et avons renseigné les trois paramètres macros : {ALERT.SENDTO}, {ALERT.SUBJECT} et {ALERT.MESSAGE}.

Nous avons défini le **Message Template** :

- Sujet : **[ALERTE] {EVENT.NAME}**
- Message : **Panne détectée sur {HOST.NAME} à {EVENT.TIME}. Sévérité : {EVENT.SEVERITY}.**

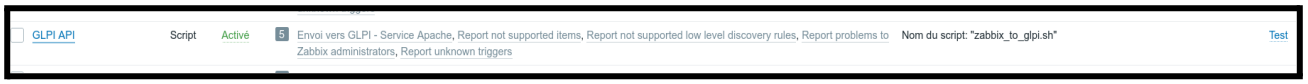


Modèles de message



Nous avons ensuite associé ce média à l'utilisateur Admin, disponible 24/7, et activé la "Trigger Action" pour que chaque passage à l'état "Problème" exécute le script automatiquement.

Comme le montre le screen dessous le média et bien activé:

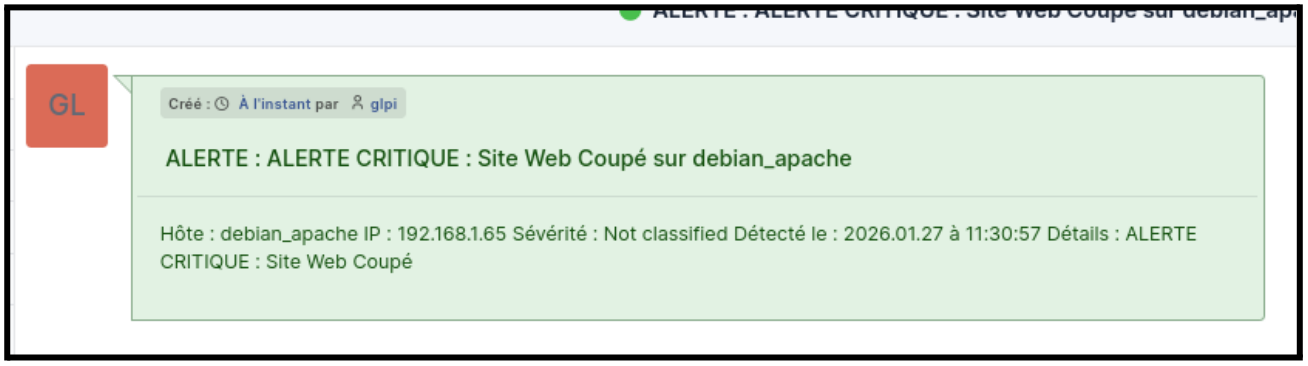


Étape 4 : Validation et recette technique

Pour vérifier que tout fonctionnait, nous avons simulé un incident sur le service Apache.
Nous avons exécuté la commande systemctl stop apache2 sur le serveur supervisé.

Nous avons constaté dans Zabbix que le trigger était passé au rouge et que la colonne **Actions** confirmait l'envoi du message via notre script.
Dans GLPI, nous avons immédiatement vu l'ouverture d'un ticket contenant toutes les informations précises sur l'incident.

Comme le montre ce screen Alerte a bien remonter dans Glpi



Et l'alerte et bien afficher aussi dans Zabbix

Conclusion

La réalisation de ce projet de supervision et de sécurité a permis de valider la mise en œuvre d'une architecture complète et cohérente, répondant aux exigences de maintien en condition opérationnelle (MCO) et de protection des actifs. Durant ces travaux pratiques, nous avons réussi à unifier des environnements hétérogènes (Debian, Windows Server, OPNsense) sous une double surveillance : celle de la disponibilité avec Zabbix et celle de la sécurité avec le SIEM Wazuh. L'étape cruciale de l'interconnexion entre Wazuh et GLPI via l'API REST démontre que la cybersécurité ne s'arrête pas à la détection, mais s'étend jusqu'à la gestion structurée des incidents selon les principes ITIL.

En simulant des incidents réels, tels que la suppression de fichiers critiques ou des attaques brute-force SSH, nous avons prouvé l'efficacité de l'automatisation : chaque menace détectée a généré une réponse proactive (Active Response pour le bannissement IP) et une documentation immédiate (création automatique de tickets P1/P2 dans GLPI). Ce TP confirme qu'une infrastructure résiliente repose sur la corrélation intelligente des logs et sur une chaîne de remontée d'alertes automatisée, réduisant drastiquement le temps de réaction face aux menaces actuelles.