AYI NEDJIMI Consultants

Wazuh SIEM/XDR Open Source : Déploiement, Configuration

Catégorie : SOC et Detection Lecture : 8 min Publié le : 08/03/2026 Auteur : Ayi NEDJIMI

Guide complet Wazuh SIEM/XDR open source : architecture, déploiement cluster, agents multi-OS, règles de détection, FIM, vulnerability detection.

2.1 Vue d'ensemble des composants

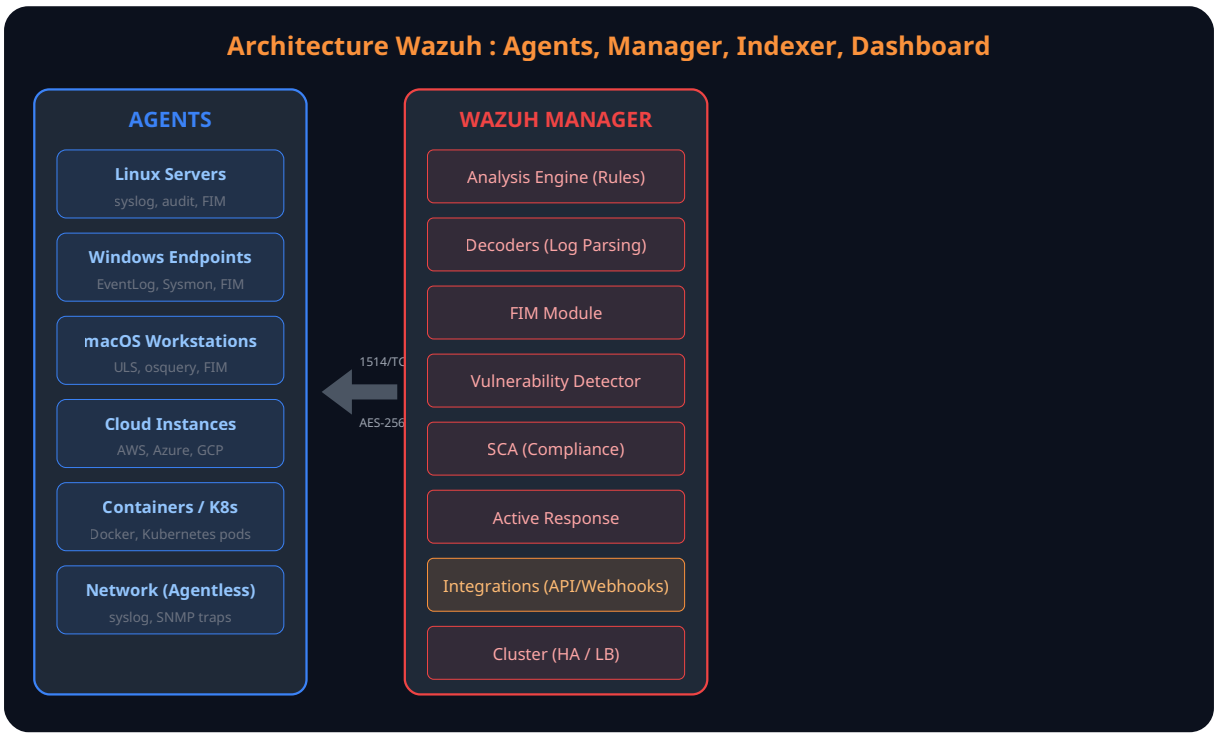
L'architecture Wazuh repose sur quatre composants principaux, chacun jouant un rôle distinct dans la chaîne de collecte, d'analyse et de visualisation des données de sécurité :

Wazuh Agent : déployé sur chaque endpoint (serveur, poste de travail, conteneur), l'agent est un processus léger qui collecte les événements système, surveille les fichiers, évalue les vulnérabilités et exécute les réponses actives. Il communique avec le manager via un canal chiffré (AES-256) sur le port 1514/TCP. L'agent supporte nativement **Linux, Windows, macOS, Solaris, AIX et HP-UX**, couvrant ainsi la quasi-totalité des environnements d'entreprise.

Wazuh Manager (Server) : c'est le cerveau de la plateforme. Le manager reçoit les données des agents, les décode via les **decoders**, les analyse via les **rules**, génère les alertes et orchestre les réponses actives. Il stocke également les configurations centralisées des agents et gère leur cycle de vie (enregistrement, mise à jour, désenregistrement). En mode cluster, plusieurs managers partagent la charge et assurent la haute disponibilité.

Wazuh Indexer : basé sur OpenSearch (fork d'Elasticsearch), l'indexer stocke et indexe toutes les alertes et les événements bruts. Il fournit les capacités de recherche full-text, d'agrégation et de rétention à long terme nécessaires aux investigations forensiques et aux exigences de conformité. Depuis la version 4.8, l'indexer supporte le **hot-warm-cold architecture** pour optimiser les coûts de stockage sur de longues périodes.

Wazuh Dashboard : basé sur OpenSearch Dashboards, il fournit l'interface graphique pour visualiser les alertes, explorer les données, gérer les agents et configurer les règles. Le dashboard inclut des modules prédéfinis pour chaque fonctionnalité (MITRE ATT&CK, Vulnerability Detection, FIM, SCA, etc.) et permet de créer des tableaux de bord personnalisés.



Votre SOC détecterait-il une attaque de type living-off-the-land ?

9200/TCP INDEXER (OpenSearch) Hot Storage Warm Storage Cold Storage Full-Text Search Aggregations DASHBOARD (OpenSearch Dash.) Alertes / Events MITRE ATT&CK Vuln Detection FIM / SCA Custom Dashboards INTEGRATIONS VirusTotal MISP TheHive / Cortex Shuffle SOAR Slack / Teams PagerDuty
Figure 1 -- Architecture Wazuh : agents, manager, indexer, dashboard et intégrations

2.2 Flux de données et chaîne de détection

Le flux de données dans Wazuh suit une chaîne logique en cinq étapes. Premièrement, l'agent collecte les événements bruts (logs système, événements Windows, modifications de fichiers, résultats de scans). Deuxièmement, ces événements sont transmis au manager via le protocole Wazuh (port 1514/TCP), chiffré en AES-256 avec compression. Troisièmement, le manager applique les **decoders** pour extraire les champs structurés des logs bruts. Quatrièmement, le moteur de règles évalue chaque événement décodé contre l'ensemble des règles actives (plus de 4 000 règles par défaut) et génère des alertes lorsqu'un match est trouvé. Cinquièmement, les alertes sont envoyées à l'indexer pour stockage et à tous les canaux d'intégration configurés (webhook, email, SOAR).

Un point crucial pour le dimensionnement : chaque agent génère en moyenne **50 à 200 événements par seconde** (EPS) selon le type de workload et la configuration du monitoring. Un manager single-node peut traiter environ 500 à 1 000 EPS avant de nécessiter un cluster. L'indexer est généralement le goulot d'étranglement, car il doit indexer, stocker et rendre searchable chaque événement.

2.3 Mode cluster : haute disponibilité

Pour les environnements de production, Wazuh supporte un mode cluster natif où plusieurs managers fonctionnent ensemble. Le cluster utilise un modèle **master-worker** : un noeud master synchronise la configuration et les clés d'agents, tandis que les noeuds workers traitent les événements. La répartition des agents entre les workers se fait automatiquement ou manuellement. En cas de panne du master, un worker peut être promu manuellement (le failover automatique du master n'est pas encore natif en 4.9, mais les workers continuent de fonctionner indépendamment).

L'indexer, basé sur OpenSearch, dispose de son propre mécanisme de clustering avec réplication des shards. Pour la production, un minimum de **3 noeuds indexer** est recommandé pour garantir la résilience (factor de réplication = 1, ce qui signifie que chaque shard a une copie sur un autre noeud).

Recommandation architecture production
Pour un déploiement de 500 à 2 000 agents : 1 manager master + 2 workers, 3 noeuds indexer, 1 dashboard. Pour 2 000 à 10 000 agents : 1 master + 4 workers, 5 noeuds indexer (3 data + 2 data-warm), 2 dashboards derrière un load balancer. Au-delà de 10 000 agents, contactez un intégrateur spécialisé ou envisagez une architecture multi-cluster avec une collecte cloud native complémentaire.

Cas concret
La détection de Log4Shell (CVE-2021-44228) a mis en évidence les limites des SIEM traditionnels. Les tentatives d'exploitation utilisaient des techniques d'obfuscation variées (${lower:j}ndi, encodage Base64) qui contournaient les signatures de détection statiques. Les SOC équipés de règles comportementales ont détecté l'exploitation 48 heures plus tôt en moyenne.

La configuration Sysmon doit être soigneusement choisie. La configuration **SwiftOnSecurity** est un excellent point de départ, mais pour les environnements à haut risque, la configuration **Olaf Hartong** (sysmon-modular) offre une couverture ATT&CK plus étendue, au prix d'un volume d'événements plus important. Quel que soit le choix, la détection des techniques d'escalade de privilèges Windows dépend directement de la qualité de cette configuration.

4.3 Configuration centralisée avec agent.conf et shared groups

L'un des atouts majeurs de Wazuh est la possibilité de gérer la configuration des agents de manière centralisée via le fichier `agent.conf` et le système de **groupes partagés**. Chaque agent peut appartenir à un ou plusieurs groupes, et hériter de la configuration de ces groupes :

```
# /var/ossec/etc/shared/linux-servers/agent.conf
<agent_config os="linux">
  <!-- Surveillance des fichiers critiques -->
  <syscheck>
    <frequency>3600</frequency>
    <directories check_all="yes" realtime="yes">/etc,/usr/bin,/usr/sbin</directories>
    <directories check_all="yes">/boot</directories>
    <ignore>/etc/mtab</ignore>
    <ignore>/etc/hosts.deny</ignore>
    <ignore type="sregex">.log$|.tmp$</ignore>
  </syscheck>

  <!-- Collecte des logs -->
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/auth.log</location>
  </localfile>

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

  <!-- Commandes de détection -->
  <localfile>
    <log_format>full_command</log_format>
    <command>netstat -tulnp</command>
    <frequency>300</frequency>
  </localfile>

  <!-- Vulnerability detection -->
  <wodle name="syscollector">
    <disabled>no</disabled>
    <interval>1h</interval>
    <packages>yes</packages>
    <hotfixes>yes</hotfixes>
    <ports all="no">yes</ports>
    <processes>yes</processes>
  </wodle>
</agent_config>
```

Pour affecter un agent à un groupe, utilisez l'API ou la CLI :

```
# Affecter un agent au groupe "linux-servers"
/var/ossec/bin/agent_groups -i 001 -g linux-servers

# Ou via l'API REST
curl -k -X PUT "https://10.0.1.20:55000/agents/001/group/linux-servers" \
  -H "Authorization: Bearer $TOKEN"
```

Une fois le décodeur en place, créons des règles de détection progressives -- du simple échec d'authentification au brute force détecté :

```
<!-- /var/ossec/etc/rules/local_rules.xml -->

<!-- Règle de base : échec d'authentification applicatif -->
<group name="custom-auth,authentication_failed,">
  <rule id="100001" level="5">
    <decoded_as>custom-auth-app</decoded_as>
    <match>FAILURE</match>
    <description>Application auth failure: user $(dstuser) from $(srcip)</description>
    <mitre>
      <id>T1110</id>
    </mitre>
    <group>authentication_failed,</group>
  </rule>

  <!-- Corrélation : 5 échecs en 120 secondes = brute force -->
  <rule id="100002" level="10" frequency="5" timeframe="120">
    <if_matched_sid>100001</if_matched_sid>
    <same_source_ip />
    <description>Brute force detected on custom app from $(srcip) targeting $(dstuser)</
description>
    <mitre>
      <id>T1110.001</id>
    </mitre>
    <group>authentication_failed,attack,</group>
  </rule>

  <!-- Corrélation avancée : brute force distribué (même user, IPs différentes) -->
  <rule id="100003" level="12" frequency="10" timeframe="300">
    <if_matched_sid>100001</if_matched_sid>
    <same_field>dstuser</same_field>
    <different_source_ip />
    <description>Distributed brute force: multiple IPs targeting user $(dstuser)</
description>
    <mitre>
      <id>T1110.004</id>
    </mitre>
    <group>authentication_failed,attack,</group>
  </rule>
</group>
```

Notez l'utilisation du mapping **MITRE ATT&CK** natif (`<mitre><id>T1110</id></mitre>`) qui permet de visualiser automatiquement les détections dans la matrice ATT&CK du dashboard Wazuh. Cette cartographie est essentielle pour structurer la couverture de détection et s'aligne avec les pratiques décrites dans notre article sur les techniques MITRE ATT&CK les plus utilisées.

5.4 Détection des techniques Living off the Land

Les attaques Living off the Land (LOLBins) exploitent des outils légitimes du système (PowerShell, WMI, certutil, regsvr32) pour échapper à la détection. Wazuh, combiné avec Sysmon, excelle dans la détection de ces techniques :

```
<!-- Détection d'utilisation suspecte de certutil pour téléchargement -->
<rule id="100010" level="12">
  <if_sid>61603</if_sid>
  <field name="win.eventdata.originalFileName">CertUtil.exe</field>
  <field name="win.eventdata.commandLine" type="pcre2">(?i)(urlcache|split|decode|encode|
download)</field>
  <description>Suspicious certutil usage: possible download/decode operation</description>
  <mitre>
    <id>T1105</id>
    <id>T1140</id>
  </mitre>
</rule>

<!-- Détection de PowerShell encodé en base64 -->
<rule id="100011" level="14">
  <if_sid>61603</if_sid>
  <field name="win.eventdata.originalFileName">PowerShell.EXE</field>
  <field name="win.eventdata.commandLine" type="pcre2">(?i)(-enc|-encodedcommand|-ec)\s+
[A-Za-z0-9+/=]{50,}</field>
  <description>PowerShell with long base64 encoded command detected</description>
  <mitre>
    <id>T1059.001</id>
    <id>T1027</id>
  </mitre>
</rule>

<!-- Détection de LOLBAS: mshta exécutant du contenu distant -->
<rule id="100012" level="13">
  <if_sid>61603</if_sid>
  <field name="win.eventdata.originalFileName">MSHTA.EXE</field>
  <field name="win.eventdata.commandLine" type="pcre2">(?i)(http|https|ftp|javascript|
vbscript)</field>
  <description>MSHTA executing remote or scripted content</description>
  <mitre>
    <id>T1218.005</id>
  </mitre>
</rule>
```

Tester les règles avant déploiement

Wazuh fournit l'outil `/var/ossec/bin/wazuh-logtest` qui permet de tester un log contre les décodeurs et règles en temps réel. Collez simplement une ligne de log, et l'outil affiche le décodeur utilisé, les champs extraits et la règle matchée. C'est indispensable pour valider les règles custom avant de les déployer en production. Exécutez `/var/ossec/bin/wazuh-logtest` et soumettez des échantillons de logs pour vérifier le comportement attendu.

```
# Extrait d'une politique SCA custom (YAML)
# /var/ossec/etc/shared/default/custom-sca-linux.yml
policy:
  id: "custom_linux_hardening"
  file: "custom-sca-linux.yml"
  name: "Custom Linux Hardening Policy"
  description: "Politique de durcissement interne"

checks:
  - id: 10001
    title: "SSH: PermitRootLogin should be disabled"
    condition: all
    rules:
      - 'f:/etc/ssh/sshd_config -> r:^\s*PermitRootLogin\s+no'

  - id: 10002
    title: "SSH: Password authentication should be disabled"
    condition: all
    rules:
      - 'f:/etc/ssh/sshd_config -> r:^\s*PasswordAuthentication\s+no'

  - id: 10003
    title: "Firewall should be active"
    condition: any
    rules:
      - 'c:systemctl is-active ufw -> r:^active'
      - 'c:systemctl is-active firewalld -> r:^active'
      - 'c:iptables -L -n -> r:Chain INPUT .+ policy DROP'

  - id: 10004
    title: "No world-writable files in /etc"
    condition: none
    rules:
      - 'c:find /etc -type f -perm -002 -> r:^/'
```

Le SCA est un outil puissant pour maintenir la conformité continue, particulièrement dans le contexte des exigences ISO 27001 et NIS 2 qui imposent une évaluation régulière de la posture de sécurité.

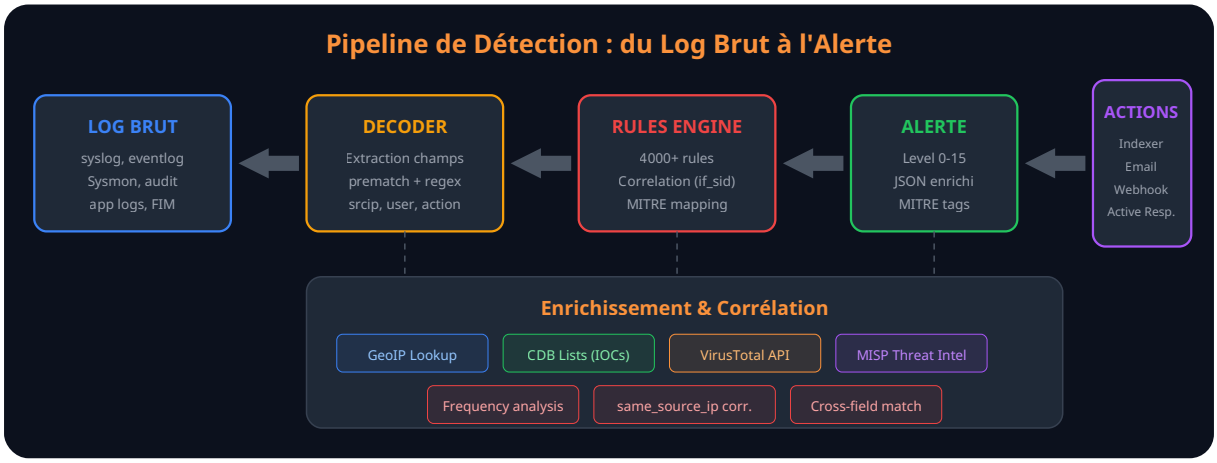


Figure 2 -- Pipeline de détection Wazuh : du log brut à l'alerte enrichie avec corrélation

Voici la checklist complète pour un déploiement Wazuh en production :

Checklist de déploiement

- **Infrastructure** : dimensionner CPU/RAM/stockage selon le nombre d'agents et la rétention cible. Prévoir la croissance à 12 mois.
- **Réseau** : ouvrir les ports 1514/TCP (agents), 1516/TCP (cluster), 55000/TCP (API), 9200/TCP (indexer), 443/TCP (dashboard). Segmenter en VLAN dédié.
- **TLS** : configurer les certificats TLS pour toutes les communications inter-composants. Ne jamais utiliser les certificats par défaut en production.
- **Cluster** : déployer au minimum 3 noeuds indexer pour la résilience. Configurer le cluster manager si > 500 agents.
- **Agents** : définir les groupes d'agents par OS et par rôle. Préparer les configurations centralisées (agent.conf) pour chaque groupe.
- **Sysmon** : déployer Sysmon sur tous les endpoints Windows avec une configuration validée (SwiftOnSecurity ou sysmon-modular).
- **Règles custom** : créer les règles de détection spécifiques à l'environnement. Mapper sur MITRE ATT&CK. Tester avec wazuh-logtest.
- **FIM** : configurer la surveillance des répertoires critiques. Activer le realtime et le who-data sur les assets sensibles.
- **Vulnerability Detection** : activer le module syscollector sur tous les agents et le vulnerability detector sur le manager.
- **SCA** : déployer les politiques CIS Benchmarks adaptées à chaque OS. Créer les politiques custom pour les exigences internes.
- **Intégrations** : configurer VirusTotal (FIM enrichment), MISP (CDB lists), TheHive (incident response). Tester le flux complet.
- **Active Response** : configurer les réponses automatiques pour les alertes critiques. Définir les listes blanches. Tester en staging.
- **Dashboards** : créer les dashboards opérationnels (SOC overview, auth monitoring, threat intel, compliance). Former l'équipe.
- **Rétention** : configurer les politiques ISM pour la gestion du cycle de vie des index (hot/warm/cold/delete).
- **Backup** : sauvegarder la configuration (/var/ossec/etc/), les règles custom, les clés d'agents et les snapshots indexer.
- **Monitoring** : surveiller la santé du cluster (charge manager, espace disque indexer, agents déconnectés). Alerter sur les anomalies.
- **Documentation** : documenter l'architecture, les règles custom, les playbooks de réponse et les procédures d'escalade.

Pour approfondir ce sujet, consultez notre outil open-source siem-correlation-rules qui facilite l'optimisation des règles de corrélation SIEM.

Questions frequentes

Comment mettre en place Wazuh SIEM/XDR Open Source dans un environnement de production ?

La mise en place de Wazuh SIEM/XDR Open Source en production necessite une planification rigoureuse, incluant l'evaluation des prerequis techniques, la definition d'une architecture cible, des tests de validation approfondis et un plan de deploiement progressif avec des points de controle a chaque etape.

Pourquoi Wazuh SIEM/XDR Open Source est-il essentiel pour la securite des systemes d'information ?

Wazuh SIEM/XDR Open Source constitue un element fondamental de la securite des systemes d'information car il permet de reduire significativement la surface d'attaque, d'ameliorer la detection des menaces et de renforcer la posture globale de securite de l'organisation face aux cybermenaces actuelles.

Combien de règles de détection faut-il pour démarrer avec Wazuh SIEM/XDR Open Source : Déploiement, Configuration ?

Commencez par 20 à 30 règles alignées sur les techniques MITRE ATT&CK les plus courantes. Mieux vaut peu de règles bien calibrées que des centaines qui génèrent du bruit.

Sources et références : MITRE ATT&CK · MITRE CAR

> Points clés à retenir
>
> • Questions frequentes
> • 13. Conclusion : Wazuh comme fondation SOC open source

13. Conclusion : Wazuh comme fondation SOC open source

Wazuh représente une opportunité unique pour les organisations de toutes tailles de déployer une **capacité de détection et réponse mature sans coûts de licence**. La plateforme couvre l'essentiel des fonctions d'un SOC moderne : collecte centralisée de logs, détection basée sur les règles, surveillance de l'intégrité, évaluation des vulnérabilités, conformité continue et réponse automatisée.

Le véritable coût de Wazuh n'est pas dans la licence (il n'y en a pas) mais dans **l'expertise nécessaire pour le déployer, le configurer et le maintenir efficacement**. Les règles par défaut fournissent une couverture de base, mais la valeur réelle vient des règles custom adaptées à l'environnement, des intégrations avec l'écosystème CTI/SOAR, et du tuning continu pour maintenir un ratio signal/bruit acceptable.

Pour aller plus loin, explorez notre article sur le Threat Hunting qui explique comment utiliser les données collectées par Wazuh pour mener des chasses proactives aux menaces. Combinée avec une méthodologie de threat hunting structurée, la plateforme Wazuh devient un véritable multiplicateur de force pour les équipes de sécurité.

Mot de la fin : Un SIEM sans analyste compétent n'est qu'un générateur de logs coûteux. Un analyste compétent sans SIEM n'est qu'un pompier sans lance. Wazuh fournit l'outil -- investissez dans les compétences pour en extraire toute la valeur.

Articles connexes

SOC & Détection
Threat Hunting : Méthodologie, Outils et Pratique
Modèle PEAK, hypothèses de chasse, outils et métriques
Techniques & Défense
MITRE ATT&CK : Top Techniques 2026 et Défense
Techniques les plus utilisées et stratégies de détection
Forensics
Forensique mémoire avec Volatility 3
Analyse mémoire RAM, détection malware, investigation
Techniques offensives
Living off the Land : binaires légitimes détournés
LOLBins, LOLDrivers et techniques d'évasion
Conformité
ISO 27001 : Guide complet
Certification, contrôles et mise en oeuvre SMSI
Attaques réseau
Attaques DNS : Tunneling, Hijacking, Poisoning
Techniques d'exploitation DNS et mesures de détection

Références et ressources externes

- Documentation officielle Wazuh 4.9 -- Guide complet d'installation, configuration et administration
- GitHub Wazuh -- Code source, issues et contributions communautaires
- Blog Wazuh -- Cas d'usage, tutoriels et nouveautés
- MITRE ATT&CK Framework -- Base de connaissances des techniques adverses
- SOCFortress Blog -- Tutoriels avancés d'intégration Wazuh

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

ayinedjimi-consultants.fr · ayi@ayinedjimi-consultants.fr