FR

Guide Pratique

SIEM avec Wazuh + OPNsense

Centraliser et corroler tous les logs -- Stack BOTUM

Mars 2026

Sommaire

1. Pourquoi un SIEM ? Le probleme des outils isoles
2. Wazuh vs ELK Stack vs Graylog
3. Installer Wazuh en Docker sur Proxmox
4. Deployer l'agent Wazuh sur OPNsense
5. Integrer les logs : Suricata, CrowdSec, AdGuard, fail2ban
6. Regles de correlation personnalisees
7. Alerting : email SMTP et Telegram webhook
8. Dashboard Wazuh : vue unifiee
9. Connexion avec Grafana (Billet 9)

Pourquoi un SIEM ? Le problème des outils isolés

Sur une infrastructure sans SIEM, vous avez des alertes partout — Suricata détecte un scan, CrowdSec bloque une IP, fail2ban interdit un autre accès SSH — mais personne ne fait le lien entre ces événements. Une attaque coordonnée peut se dérouler sur 3 heures, à travers 5 outils différents, sans que vous voyiez jamais la séquence complète.

Un **SIEM** (Security Information and Event Management) résout exactement ça : il collecte les logs de *toutes* les sources, les normalise, les corrèle, et déclenche une alerte quand un pattern d'attaque se dessine. C'est la différence entre avoir des caméras de surveillance dans chaque pièce et avoir un agent de sécurité qui surveille tous les écrans en même temps.

Wazuh vs ELK Stack vs Graylog

J'ai évalué trois options pour ce stack :

• **ELK Stack (Elasticsearch + Logstash + Kibana) :** puissant, flexible, mais vorace en RAM (8-16 Go minimum) et nécessite de créer toutes les règles manuellement.

• **Graylog :** bonne interface, plus léger qu'ELK, mais les règles de corrélation sont limitées en version gratuite.

• **Wazuh :** léger (4 Go RAM), 2000+ règles de détection intégrées, agent natif FreeBSD pour OPNsense, compliance PCI DSS / CIS natif, et dashboard inclus. **Mon choix.**

Installer Wazuh en Docker sur Proxmox

Je déploie Wazuh sur une VM Proxmox dédiée : 2 vCPU, 4 Go RAM, 40 Go SSD. La stack Docker Compose officielle Wazuh inclut trois containers :

• **wazuh-manager :** le moteur de corrélation et d'analyse des logs

• **wazuh-indexer :** OpenSearch (fork d'Elasticsearch) pour stocker et indexer les alertes

• **wazuh-dashboard :** interface web (port 443)

```
# VM Proxmox -- Debian/Ubuntu
# Ports à ouvrir depuis OPNsense : 1514/UDP, 1515/TCP, 55000/TCP, 443/TCP

mkdir -p /opt/wazuh && cd /opt/wazuh

# Télécharger le docker-compose officiel
curl -sSL https://raw.githubusercontent.com/wazuh/wazuh-docker/v4.9.2/single-node/docker-compo
  se.yml      -o docker-compose.yml

# Générer les certificats SSL internes
docker compose -f generate-indexer-certs.yml run --rm generator

# Lancer la stack complète
docker compose up -d

# Vérification
docker compose ps
# wazuh-manager   Up   0.0.0.0:1514->1514/udp, 1515/tcp, 55000/tcp
# wazuh-indexer   Up   9200/tcp
# wazuh-dashboard Up   0.0.0.0:443->5601/tcp
```

Accédez au dashboard sur `https://IP-PROXMOX` avec les credentials par défaut `admin / SecretPassword` — à changer immédiatement.

Déployer l'agent Wazuh sur OPNsense

OPNsense tourne sur FreeBSD — l'agent Wazuh FreeBSD est compatible nativement. Installation via SSH :

```
# Connexion SSH à OPNsense
ssh admin@192.168.1.1

# Installer l'agent Wazuh FreeBSD
pkg install -y wazuh-agent

# Configurer l'adresse du manager
sed -i '' 's|<address>MANAGER_IP</address>|<address>192.168.X.X</address>|'     /var/ossec/etc
  /ossec.conf

# Activer et démarrer l'agent
echo 'wazuh_agent_enable="YES"' >> /etc/rc.conf
service wazuh-agent start

# Vérifier la connexion au manager
/var/ossec/bin/agent_control -i 001
```

Configuration ossec.conf sur OPNsense

```
<ossec_config>
  <client>
    <server>
      <address>192.168.X.X</address>
      <port>1514</port>
      <protocol>udp</protocol>
    </server>
    <enrollment>
      <enabled>yes</enabled>
      <manager_address>192.168.X.X</manager_address>
      <port>1515</port>
      <agent_name>opnsense-fw</agent_name>
    </enrollment>
  </client>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/system.log</location>
  </localfile>
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/filter.log</location>
  </localfile>
</ossec_config>
```

Intégrer les logs : Suricata, CrowdSec, AdGuard, fail2ban

Suricata EVE JSON (Billet 7)

Activer EVE JSON dans OPNsense : **Services > Intrusion Detection > Administration** → EVE output enabled, types : alert, dns, http, tls, flow.

```
<!-- ossec.conf agent OPNsense -->
<localfile>
  <log_format>json</log_format>
  <location>/var/log/suricata/eve.json</location>
  <label key="source">suricata</label>
</localfile>
```

CrowdSec (Billet 5)

```
<localfile>
  <log_format>json</log_format>
  <location>/var/log/crowdsec.log</location>
  <label key="source">crowdsec</label>
</localfile>

# Webhook CrowdSec -> Wazuh (alertes instantanées)
# /etc/crowdsec/notifications/http.yaml :
type: http
name: wazuh_notifier
url: http://192.168.X.X:55000/
method: POST
template: |
  {"source":"crowdsec","action":"{{.Decision.Type}}","ip":"{{.Decision.Value}}"}
```

AdGuard Home DNS (Billet 8) et fail2ban

```
<!-- AdGuard Home : querylog.json -->
<localfile>
  <log_format>json</log_format>
  <location>/opt/adguardhome/work/data/querylog.json</location>
  <label key="source">adguard</label>
</localfile>

<!-- fail2ban (host Linux) -->
<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/fail2ban.log</location>
  <label key="source">fail2ban</label>
</localfile>

<!-- Syslog OPNsense -> manager : System > Log Files > Settings
     Remote syslog : 192.168.X.X:514/UDP -->
```

Règles de corrélation personnalisées

Les règles de corrélation Wazuh permettent de détecter des séquences d'attaque multi-étapes. Voici le fichier de règles BOTUM :

```
<!-- /var/ossec/etc/rules/botum_correlation.xml -->
<group name="botum_correlation,">

  <!-- Règle 1 : Suricata + CrowdSec (même IP, fenêtre 5 min) -->
  <rule id="100100" level="12" timeframe="300" ignore="60">
    <if_matched_sid>86601</if_matched_sid>
    <same_field>data.src_ip</same_field>
    <description>CRITIQUE: $(data.src_ip) -- Suricata+CrowdSec simultanés</description>
    <options>alert_by_email</options>
    <group>attack_correlation,critical</group>
  </rule>

  <!-- Règle 2 : Brute force SSH (5 tentatives / 2 min) -->
  <rule id="100101" level="10" frequency="5" timeframe="120">
    <if_matched_sid>5710</if_matched_sid>
    <same_field>data.srcip</same_field>
    <description>Brute force SSH : $(data.srcip) -- $(frequency)x/2min</description>
    <group>brute_force,authentication</group>
  </rule>

  <!-- Règle 3 : Exfiltration DNS (100 queries/min même client) -->
  <rule id="100102" level="8" frequency="100" timeframe="60">
    <if_matched_sid>82200</if_matched_sid>
    <same_field>data.client</same_field>
    <description>DNS anormal : $(frequency) queries/min -- $(data.client)</description>
    <group>dns_anomaly,data_exfiltration</group>
  </rule>

  <!-- Règle 4 : IP récidiviste fail2ban (3 bans / 1h) -->
  <rule id="100103" level="9" frequency="3" timeframe="3600">
    <if_matched_sid>0504</if_matched_sid>
    <same_field>data.srcip</same_field>
    <description>Récidiviste : $(data.srcip) bannie $(frequency)x en 1h</description>
    <group>recidivist,fail2ban</group>
  </rule>

</group>
```

```
# Déployer les règles personnalisées
docker cp botum_correlation.xml wazuh-manager:/var/ossec/etc/rules/
docker exec wazuh-manager /var/ossec/bin/wazuh-logtest   # Vérifier syntaxe
docker exec wazuh-manager /var/ossec/bin/wazuh-control restart
```

Alerting : email et Telegram webhook

Configuration email SMTP

```
<ossec_config>
  <global>
    <email_notification>yes</email_notification>
    <smtp_server>mail.botum.ca</smtp_server>
    <email_from>wazuh@botum.ca</email_from>
    <email_to>admin@votredomaine.com</email_to>
    <email_maxperhour>20</email_maxperhour>
  </global>
  <alerts>
    <log_alert_level>3</log_alert_level>
    <email_alert_level>10</email_alert_level>
  </alerts>
</ossec_config>
```

Webhook Telegram (alertes mobiles en temps réel)

```
#!/bin/bash
# /var/ossec/integrations/custom-telegram
ALERT_FILE=$1
BOT_TOKEN="VOTRE_BOT_TOKEN"
CHAT_ID="VOTRE_CHAT_ID"

LEVEL=$(python3 -c "import json; d=json.load(open('$ALERT_FILE')); print(d['rule']['level'])")
DESC=$(python3 -c "import json; d=json.load(open('$ALERT_FILE')); print(d['rule']['description
  '])")
AGENT=$(python3 -c "import json; d=json.load(open('$ALERT_FILE')); print(d['agent']['name'])")

MSG="? *WAZUH -- Niveau $LEVEL*%0A? $DESC%0A?? $AGENT"
curl -s -X POST "https://api.telegram.org/bot${BOT_TOKEN}/sendMessage"      -d "chat_id=${CHAT
  _ID}&text=${MSG}&parse_mode=Markdown" > /dev/null

# Activer dans ossec.conf :
# <integration>
#   <name>custom-telegram</name>
#   <level>10</level>
#   <alert_format>json</alert_format>
# </integration>
```

Dashboard Wazuh : vue unifiée

Le dashboard Wazuh centralise tous les événements du stack en une seule interface. Modules clés à activer :

- **Security Events :** tous les événements corrélés en temps réel
- **Threat Intelligence :** IOC (Indicators of Compromise) via feeds externes
- **Integrity Monitoring :** surveillance des fichiers système OPNsense
- **Vulnerability Detection :** CVE sur les paquets installés (Wazuh SCA)
- **Compliance PCI/CIS :** rapports automatiques de conformité
- **MITRE ATT&CK :** cartographie des incidents sur la matrice MITRE

Connexion avec Grafana (Billet 9)

Wazuh (sécurité) + Grafana (métriques réseau) = la combinaison parfaite. Je connecte les deux via **OpenSearch comme source commune** dans Grafana :

```
# Grafana -> Connections -> Data Sources -> Add -> OpenSearch
# Host : https://192.168.X.X:9200
# Index : wazuh-alerts-4.x-*
# Time field : @timestamp
# Version : OpenSearch 2.x
# Auth : admin / VotreMotDePasse

# Annotations Grafana depuis Wazuh :
# Dashboard -> Settings -> Annotations -> Add
# Data source : OpenSearch (Wazuh)
# Query : rule.level:>=10 AND agent.name:"opnsense-fw"
# Résultat : alertes critiques affichées en barres verticales rouges
# sur les graphiques de trafic InfluxDB.
# Corrélation immédiate : pic de trafic -> alerte Suricata -> ban CrowdSec.
```

Conclusion et prochaines étapes

Avec ce Billet 11, le stack BOTUM dispose d'une **couche SIEM complète** : tous les outils de sécurité des billets précédents — Suricata, CrowdSec, AdGuard Home, fail2ban — sont maintenant correlés dans une vue unifiée, avec alertes automatiques email et Telegram.

La sécurité de l'infrastructure BOTUM est maintenant composée de :

• Billet 7 — Suricata IDS/IPS : détection et prévention d'intrusions

• Billet 5 — CrowdSec + Fail2ban : protection collaborative

• Billet 8 — AdGuard Home : filtrage DNS

• Billet 9 — Grafana + InfluxDB : monitoring réseau

• Billet 10 — CARP : haute disponibilité

• **Billet 11 — Wazuh SIEM (ce guide)** : centralisation et corrélation des logs

Prochain billet (12) : Ansible — automatiser l'intégralité du déploiement de ce stack. Zéro configuration manuelle, tout versionné dans Git, déploiement d'un firewall neuf en 10 minutes.

Consultez le hub de la série Stack OPNsense pour tous les billets.

Téléchargez le guide PDF : Guide PDF

Article complet : blog.botum.ca/opnsense-wazuh-siem-centralisation-logs

Site web : www.botum.ca • contact@botum.ca