LibreNMS 2026 : Guide Complet Installation, Configuration et Intégration avec Wazuh, Suricata et Graylog

LibreNMS est la solution de supervision réseau open source la plus adoptée dans les environnements d'entreprise et de SOC (Security Operations Center) en 2026. Héritier direct d'Observium, ce fork communautaire propulsé par PHP/Laravel offre une télémétrie complète via SNMP v3, ICMP, sFlow et NetFlow, une interface web réactive, un moteur d'alertes hautement configurable et une intégration native avec les outils de sécurité leaders du marché — Wazuh, Suricata et Graylog. Ce guide exhaustif vous accompagne depuis l'installation sur Ubuntu 22.04/24.04 ou Debian 12 jusqu'à la construction d'un SOC open source complet, en passant par la configuration SNMP v3 sécurisée, les règles d'alerte, les dashboards et la sécurisation de votre infrastructure de supervision.

1. Présentation de LibreNMS

LibreNMS est né en 2013 d'un fork d'Observium, lui-même un projet de supervision réseau créé en 2006 par Adam Armstrong. Lorsque le modèle de licence d'Observium a évolué vers un modèle plus restrictif, la communauté open source a décidé de créer un fork entièrement libre sous licence GPLv3. Depuis lors, LibreNMS a connu une croissance exponentielle : plus de 650 contributeurs actifs sur GitHub, plus de 35 000 étoiles, et des centaines de mises à jour par an.

1.1 Historique et positionnement

LibreNMS se positionne comme une alternative open source complète aux solutions commerciales comme SolarWinds NPM, PRTG ou Cisco Prime Infrastructure. Contrairement à ces solutions propriétaires dont les coûts de licence peuvent atteindre plusieurs dizaines de

milliers d'euros annuels, LibreNMS offre des fonctionnalités équivalentes sans frais de licence. Son écosystème de plugins, d'intégrations et de contributions communautaires en fait l'outil de référence pour les équipes réseau et sécurité souhaitant une visibilité complète sur leur infrastructure.

Les fonctionnalités clés de LibreNMS incluent :

Auto-découverte réseau : détection automatique des équipements via SNMP, CDP, LLDP, BGP et autres protocoles

Support multi-vendor : Cisco, Juniper, HP/Aruba, Fortinet, Palo Alto, Mikrotik, VMware et plus de 400 fabricants

Surveillance temps réel : graphiques RRDtool avec métriques toutes les 5 minutes (configurable)

Moteur d'alertes avancé : conditions complexes, templates, escalades, intégrations multiples

API REST : intégration avec ITSM, CMDB, outils DevOps

Distributed polling : architecture hautement disponible pour les grands réseaux

Weathermaps : cartes réseau visuelles avec indicateurs de charge en temps réel

1.2 Architecture technique

LibreNMS repose sur une architecture PHP/Laravel avec MySQL ou MariaDB comme backend de données relationnelles, et RRDtool pour le stockage des métriques temporelles. L'interface web est servie par Nginx ou Apache, et le polling des équipements est assuré par des processus PHP planifiés via cron ou un daemon de polling.

Composant	Technologie	Rôle
Frontend	PHP 8.2+ / Laravel	Interface web, API REST
Base de données	MySQL 8.0 / MariaDB 10.6+	Stockage config, alertes, topologie
Métriques	RRDtool	Séries temporelles, graphiques
Collecte	SNMP v1/v2c/v3, ICMP, sFlow	Polling équipements réseau
Web server	Nginx / Apache	Reverse proxy, TLS termination
Alerting	LibreNMS Alert Engine	Règles, templates, transports

1.3 Licence et communauté

LibreNMS est distribué sous licence GPLv3, ce qui garantit la liberté d'utilisation, de modification et de redistribution. Le projet est hébergé sur GitHub à l'adresse `github.com/librenms/librenms` et bénéficie d'une communauté active via le forum communautaire, le canal Discord et les issues GitHub. Les mises à jour sont publiées régulièrement via le script `daily.sh` qui peut être automatisé via cron pour maintenir l'installation à jour sans intervention manuelle.

---

2. Prérequis et Architecture de Déploiement

Avant de commencer l'installation de LibreNMS, il est essentiel de comprendre l'architecture cible et de s'assurer que les prérequis matériels et logiciels sont satisfaits. Une installation correctement dimensionnée garantira des performances optimales et une supervision fiable de votre infrastructure réseau.

2.1 Prérequis matériels

Taille réseau	CPU	RAM	Stockage	Devices max
Petit	2 vCPU	4 GB	50 GB SSD	~100
Moyen	4 vCPU	8 GB	200 GB SSD	~500
Grand	8 vCPU	16 GB	500 GB SSD	~2000
Distribué	16+ vCPU	32+ GB	1+ TB SSD	10000+

2.2 Prérequis logiciels

LibreNMS nécessite les composants logiciels suivants pour fonctionner correctement :

Système d'exploitation : Ubuntu 22.04 LTS, Ubuntu 24.04 LTS ou Debian 12 (Bookworm)

PHP : 8.2 ou 8.3 (avec extensions : bcmath, curl, gd, gmp, json, mbstring, memcached, openssl, pdo, snmp, xml, zip)

MySQL : 8.0+ ou MariaDB 10.6+

Web server : Nginx 1.18+ (recommandé) ou Apache 2.4+

Python : 3.8+ (pour les scripts de supervision)

Net-SNMP : 5.9+ (pour le polling SNMP)

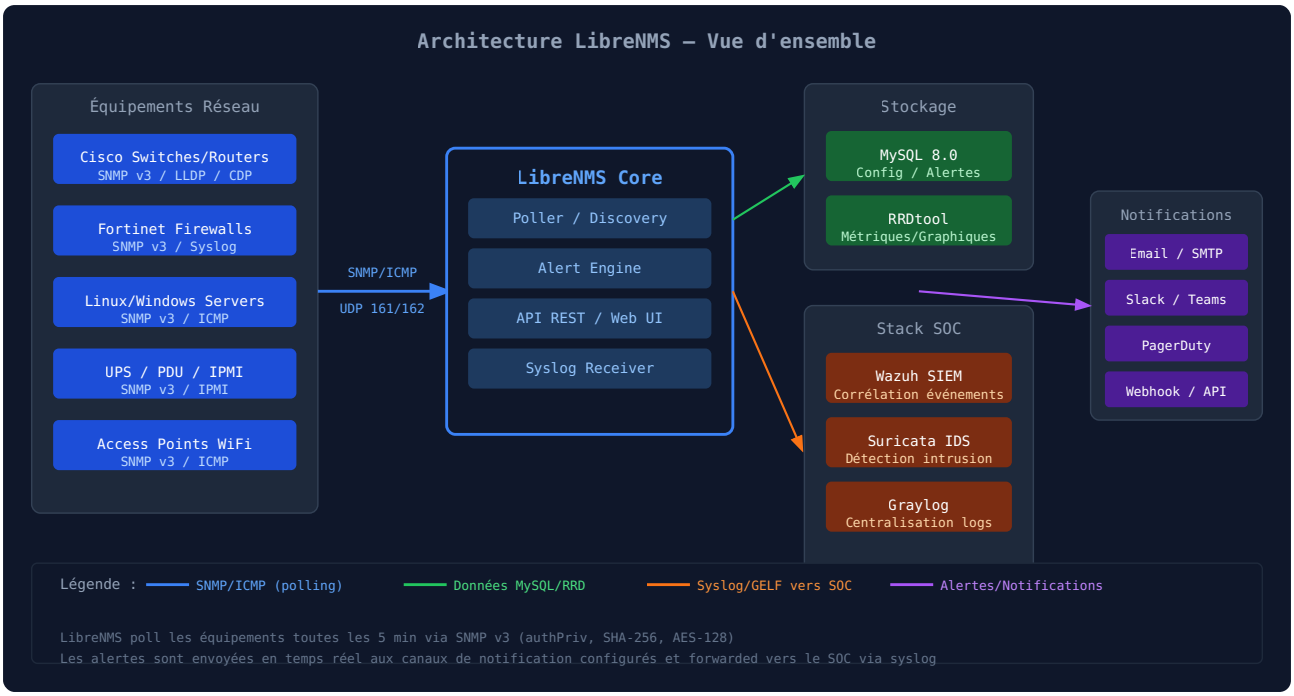
RRDtool : 1.7+ (pour les graphiques)

fping : 5.0+ (pour les vérifications ICMP)

Composer : 2.x (gestionnaire de dépendances PHP)

2.3 Architecture réseau recommandée

Dans un déploiement de production, LibreNMS doit être isolé dans un VLAN de management dédié. Les équipements supervisés n'exposent leur interface SNMP que vers ce VLAN, minimisant la surface d'attaque. Le serveur LibreNMS dispose d'accès sortants vers les équipements supervisés sur UDP 161 (SNMP) et ICMP, et reçoit des traps SNMP sur UDP 162.



3. Installation Pas à Pas sur Ubuntu 22.04

L'installation de LibreNMS sur Ubuntu 22.04 LTS suit un processus bien documenté par la communauté. Nous allons suivre les étapes officielles en ajoutant des précisions et bonnes pratiques pour un déploiement en production sécurisé.

3.1 Préparation du système

Commencez par mettre à jour le système et installer les dépendances de base :

```
 # Mise à jour système
apt update && apt upgrade -y

# Installation des dépendances système
apt install -y acl curl fping git graphviz imagemagick mariadb-client \
    mariadb-server mtr-tiny nginx nmap php8.2-cli php8.2-curl \
    php8.2-fpm php8.2-gd php8.2-gmp php8.2-mbstring php8.2-mysql \
    php8.2-snmp php8.2-xml php8.2-zip rrdtool snmp snmpd whois \
    python3-pymysql python3-dotenv python3-redis python3-setuptools \
    python3-systemd python3-pip unzip traceroute

# Installation de Composer
curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=compos
```

3.2 Création de l'utilisateur librenms

LibreNMS doit fonctionner sous un utilisateur système dédié pour des raisons de sécurité :

```
 # Création de l'utilisateur librenms (sans shell de connexion interactif)
useradd librenms -d /opt/librenms -M -r -s "$(which bash)"

# Clonage du dépôt LibreNMS
cd /opt
git clone https://github.com/librenms/librenms.git

# Permissions
chown -R librenms:librenms /opt/librenms
chmod 771 /opt/librenms
setfacl -d -m g::rwx /opt/librenms/rrd /opt/librenms/logs \
    /opt/librenms/bootstrap/cache/ /opt/librenms/storage/
setfacl -R -m g::rwx /opt/librenms/rrd /opt/librenms/logs \
    /opt/librenms/bootstrap/cache/ /opt/librenms/storage/
```

3.3 Installation des dépendances PHP via Composer

```
 # Installation des dépendances PHP (en tant qu'utilisateur librenms)
su -s /bin/bash librenms -c "./scripts/composer_wrapper.php install --no-dev"

# Si la commande échoue, utiliser directement Composer
su -s /bin/bash librenms -c "composer install --no-dev"
```

3.4 Configuration de MariaDB/MySQL

LibreNMS nécessite une configuration spécifique de MySQL pour fonctionner correctement, notamment l'activation du mode strict et l'encodage UTF-8 :

```
 # Sécurisation de l'installation MariaDB
mysql_secure_installation

# Connexion à MySQL pour créer la base de données
mysql -u root -p << 'EOF'
CREATE DATABASE librenms CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
CREATE USER 'librenms'@'localhost' IDENTIFIED BY 'VotreMotDePasseSecurise2026!';
GRANT ALL PRIVILEGES ON librenms.* TO 'librenms'@'localhost';
FLUSH PRIVILEGES;
EXIT;
EOF

# Configuration MySQL (/etc/mysql/mariadb.conf.d/50-server.cnf)
cat >> /etc/mysql/mariadb.conf.d/50-server.cnf << 'EOF'

[mysqld]
innodb_file_per_table=1
lower_case_table_names=0
character-set-server = utf8mb4
collation-server = utf8mb4_unicode_ci
EOF

systemctl restart mariadb
```

3.5 Configuration PHP

Ajustez les paramètres PHP pour LibreNMS dans le fichier de configuration PHP-FPM :

```
 # Timezone PHP (dans /etc/php/8.2/fpm/php.ini et /etc/php/8.2/cli/php.ini)
sed -i 's/;date.timezone =/date.timezone = Europe\/Paris/' /etc/php/8.2/fpm/php.ini
sed -i 's/;date.timezone =/date.timezone = Europe\/Paris/' /etc/php/8.2/cli/php.ini

# Paramètres de performance (optionnel mais recommandé)
sed -i 's/memory_limit = 128M/memory_limit = 512M/' /etc/php/8.2/fpm/php.ini
sed -i 's/upload_max_filesize = 2M/upload_max_filesize = 64M/' /etc/php/8.2/fpm/php.ini
sed -i 's/max_execution_time = 30/max_execution_time = 300/' /etc/php/8.2/fpm/php.ini
```

3.6 Configuration de Nginx

Créez un vhost Nginx dédié pour LibreNMS :

```
 cat > /etc/nginx/sites-available/librenms.conf << 'EOF'
server {
    listen 443 ssl http2;
    server_name librenms.votre-domaine.fr;

    ssl_certificate /etc/ssl/certs/librenms.crt;
    ssl_certificate_key /etc/ssl/private/librenms.key;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256;
    ssl_prefer_server_ciphers off;
    add_header Strict-Transport-Security "max-age=63072000; includeSubDomains; preload";
    add_header X-Frame-Options DENY;
    add_header X-Content-Type-Options nosniff;
    add_header Content-Security-Policy "default-src 'self'; script-src 'self' 'unsafe-inline'; sty

    root /opt/librenms/html;
    index index.php;

    charset utf-8;
    gzip on;
    gzip_types text/css application/javascript text/javascript application/x-javascript image/svg+

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ [^/]\.php(/|$) {
        fastcgi_pass unix:/var/run/php/php8.2-fpm.sock;
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        include fastcgi.conf;
    }

    location ~ /\.(?!well-known).* {
        deny all;
    }
}

server {
    listen 80;
    server_name librenms.votre-domaine.fr;
    return 301 https://$server_name$request_uri;
}
EOF

ln -s /etc/nginx/sites-available/librenms.conf /etc/nginx/sites-enabled/
nginx -t && systemctl reload nginx
```

3.7 Configuration de PHP-FPM

```
 cat > /etc/php/8.2/fpm/pool.d/librenms.conf << 'EOF'
[librenms]
user = librenms
group = librenms
listen = /var/run/php/php8.2-fpm.sock
listen.owner = www-data
listen.group = www-data
pm = dynamic
pm.max_children = 10
pm.start_servers = 2
pm.min_spare_servers = 1
pm.max_spare_servers = 5
env[HOSTNAME] = $HOSTNAME
env[PATH] = /usr/local/bin:/usr/bin:/bin
EOF

# Supprimer le pool par défaut
rm /etc/php/8.2/fpm/pool.d/www.conf

systemctl restart php8.2-fpm
```

3.8 Initialisation de LibreNMS

```
 # Copie du fichier de configuration
cp /opt/librenms/.env.example /opt/librenms/.env

# Édition de la configuration DB
cat > /opt/librenms/.env << 'EOF'
APP_URL=https://librenms.votre-domaine.fr
DB_HOST=127.0.0.1
DB_DATABASE=librenms
DB_USERNAME=librenms
DB_PASSWORD=VotreMotDePasseSecurise2026!
# SNMP trap receiver
SNMP_TRAP_HOST=127.0.0.1
# Syslog
SYSLOG_ENABLE=true
EOF

# Migration de la base de données (en tant qu'utilisateur librenms)
su -s /bin/bash librenms -c "cd /opt/librenms && php artisan migrate --force"

# Création du premier utilisateur admin
su -s /bin/bash librenms -c "cd /opt/librenms && php artisan make:user"

# Génération de la clé d'application
su -s /bin/bash librenms -c "cd /opt/librenms && php artisan key:generate"
```

3.9 Configuration de cron et logrotate

```
 # Cron pour LibreNMS
cp /opt/librenms/dist/librenms.cron /etc/cron.d/librenms

# Logrotate
cp /opt/librenms/misc/librenms.logrotate /etc/logrotate.d/librenms

# Démarrage du daemon de polling (recommandé vs cron polling)
cp /opt/librenms/dist/librenms-scheduler.service /etc/systemd/system/
systemctl enable librenms-scheduler.service
systemctl start librenms-scheduler.service

# Service dispatcher (gestion des alertes en temps réel)
cp /opt/librenms/dist/librenms-dispatch-events.service /etc/systemd/system/
systemctl enable librenms-dispatch-events.service
systemctl start librenms-dispatch-events.service
```

3.10 Validation de l'installation

LibreNMS fournit un script de validation qui vérifie tous les prérequis et la configuration :

```
 # Validation complète de l'installation
su -s /bin/bash librenms -c "cd /opt/librenms && php validate.php"

# Sortie attendue (exemple) :
# ====================================
# Component | Version
# --------- | -------
# LibreNMS  | 24.x.x
# DB Schema | 308 (OK)
# PHP       | 8.2.x
# PHP Ext   | OK
# Nginx     | 1.18.x
# RRDTool   | 1.7.x
# SNMP      | NET-SNMP 5.9.x
# ====================================
# [OK] Composer dependencies are up to date
# [OK] Database connection successful
# [OK] Database schema is current
```

---

4. Configuration SNMP v3

SNMP (Simple Network Management Protocol) est le protocole central de LibreNMS pour collecter les métriques des équipements réseau. La version 3 est obligatoire en production car elle offre authentification et chiffrement, contrairement aux versions 1 et 2c qui transmettent les community strings en clair.

4.1 Comprendre SNMP v3

SNMP v3 introduit trois niveaux de sécurité :

noAuthNoPriv : Pas d'authentification, pas de chiffrement (équivalent SNMP v1/v2c — ne jamais utiliser)

authNoPriv : Authentification via MD5 ou SHA, pas de chiffrement (insuffisant pour production)

authPriv : Authentification + chiffrement des données (recommandé en production)

Pour LibreNMS, nous utiliserons systématiquement **authPriv** avec **SHA-256** pour l'authentification et **AES-128** pour le chiffrement.

4.2 Configuration SNMP v3 sur l'agent supervisé (Linux)

```
 # Installation de l'agent SNMP
apt install snmpd -y

# Arrêt du service pour modification
systemctl stop snmpd

# Backup de la config originale
cp /etc/snmp/snmpd.conf /etc/snmp/snmpd.conf.bak

# Configuration SNMP v3 (/etc/snmp/snmpd.conf)
cat > /etc/snmp/snmpd.conf << 'EOF'
# Écoute uniquement sur le VLAN management
agentAddress  udp:192.168.10.1:161

# Informations système
sysLocation "Datacenter Paris - Rack A3"
sysContact "noc@votre-entreprise.fr"
sysName "serveur-prod-01"

# Désactiver SNMP v1/v2c (community strings)
# Ne pas définir de rocomm ou rwcomm !

# Créer l'utilisateur SNMPv3
# Format : createUser
createUser librenmsv3 SHA-256 "VotrePassAuth256Secure!" AES "VotrePassPrivAESSecure!"

# Accès en lecture seule pour cet utilisateur
rouser librenmsv3 priv

# Vues (limiter ce qui est accessible)
view systemview included .1.3.6.1.2.1.1
view systemview included .1.3.6.1.2.1.25.1.1
view systemview included .1.3.6.1.4.1

# Désactiver le proxy et forward
master agentx
agentXSocket /var/agentx/master
EOF

# Redémarrage du service SNMP
systemctl start snmpd
systemctl enable snmpd

# Test de la configuration SNMPv3
snmpget -v3 -l authPriv -u librenmsv3 -a SHA-256 -A "VotrePassAuth256Secure!" \
    -x AES -X "VotrePassPrivAESSecure!" 192.168.10.1 .1.3.6.1.2.1.1.5.0
```

4.3 Configuration SNMP v3 sur équipements Cisco

```
 ! Configuration IOS/IOS-XE pour SNMP v3
! ========================================

! Désactiver SNMP v1/v2c si encore actif
no snmp-server community public
no snmp-server community private

! Créer un groupe SNMPv3 avec authPriv
snmp-server group LibreNMSGroup v3 priv

! Créer l'utilisateur SNMPv3
! (SHA pour auth, AES128 pour priv)
snmp-server user librenmsv3 LibreNMSGroup v3 auth sha VotrePassAuth256Secure! priv aes 128 VotreP

! Définir une vue pour limiter l'accès
snmp-server view LibreNMSView iso included

! Associer la vue au groupe
snmp-server group LibreNMSGroup v3 priv read LibreNMSView

! Restreindre l'accès à l'IP du serveur LibreNMS
snmp-server host 192.168.10.50 version 3 priv librenmsv3

! SNMP traps vers LibreNMS (optionnel)
snmp-server enable traps
snmp-server host 192.168.10.50 traps version 3 priv librenmsv3

! Vérification
show snmp user
show snmp group
```

4.4 Configuration SNMP v3 sur Fortinet FortiGate

```
 # Configuration FortiOS pour SNMP v3
config system snmp sysinfo
    set status enable
    set description "FortiGate Production"
    set contact-info "noc@votre-entreprise.fr"
    set location "Datacenter Paris"
end

config system snmp user
    edit "librenmsv3"
        set security-level auth-priv
        set auth-proto sha256
        set auth-pwd VotrePassAuth256Secure!
        set priv-proto aes128
        set priv-pwd VotrePassPrivAESSecure!
        set notify-hosts 192.168.10.50
        set queries enable
        set query-port 161
        set status enable
        set trap-status enable
        set trap-v3-rport 162
    next
end
```

4.5 Ajout de devices dans LibreNMS

Après avoir configuré SNMP v3 sur vos équipements, ajoutez-les à LibreNMS via la CLI ou l'interface web :

```
 # Ajout via CLI (méthode recommandée pour les déploiements massifs)
cd /opt/librenms

# Ajout d'un serveur Linux
sudo -u librenms php addhost.php 192.168.10.1 v3 librenmsv3 VotrePassAuth256Secure! sha256 AES Vo

# Ajout d'un switch Cisco
sudo -u librenms php addhost.php 192.168.1.254 v3 librenmsv3 VotrePassAuth256Secure! sha256 AES V

# Ajout en masse depuis un fichier CSV (script custom)
while IFS=',' read -r hostname ip; do
    sudo -u librenms php addhost.php "$ip" v3 librenmsv3 VotrePassAuth256Secure! sha256 AES Votre
    echo "Ajouté: $hostname ($ip)"
done < /opt/librenms/devices.csv

# Forcer une découverte immédiate
sudo -u librenms php discovery.php -h all
```

4.6 Configuration des modules de découverte

LibreNMS propose des modules de découverte modulaires. Activez ceux pertinents pour votre infrastructure :

```
 # /opt/librenms/config/config.php (ou via l'interface Admin > Settings)
# Modules de découverte recommandés

# Topologie réseau
$config['discovery_modules']['lldp']  = true;  // Link Layer Discovery Protocol
$config['discovery_modules']['cdp']   = true;  // Cisco Discovery Protocol
$config['discovery_modules']['ospf']  = true;  // OSPF topology
$config['discovery_modules']['bgp']   = true;  // BGP peers

# Inventaire hardware
$config['discovery_modules']['entity-physical']  = true;  // Modules physiques
$config['discovery_modules']['processors']       = true;  // CPUs
$config['discovery_modules']['mempools']         = true;  // Mémoire
$config['discovery_modules']['storage']          = true;  // Stockage
$config['discovery_modules']['hr-device']        = true;  // HR-MIB

# Réseau
$config['discovery_modules']['ipv4-addresses']   = true;  // Adresses IPv4
$config['discovery_modules']['ipv6-addresses']   = true;  // Adresses IPv6
$config['discovery_modules']['ports']            = true;  // Interfaces réseau
$config['discovery_modules']['vlans']            = true;  // VLANs

# Sécurité et monitoring
$config['discovery_modules']['nac']              = true;  // Network Access Control
$config['discovery_modules']['arp-table']        = true;  // Table ARP

# MIBs custom (pour fabricants spécifiques)
$config['mibs_dir'] = ['/opt/librenms/mibs', '/opt/librenms/mibs/vendor'];
```

4.7 MIBs personnalisées

Pour superviser des équipements avec des MIBs propriétaires, LibreNMS permet d'ajouter des fichiers MIB supplémentaires :

```
 # Répertoire des MIBs
ls /opt/librenms/mibs/
# Cisco/, Juniper/, HP/, Fortinet/, etc.

# Ajouter une MIB propriétaire (exemple Schneider Electric UPS)
cp SCHNEIDER-MIB.txt /opt/librenms/mibs/vendor/

# Vérifier que la MIB est reconnue
su -s /bin/bash librenms -c "snmptranslate -M /opt/librenms/mibs/vendor -m SCHNEIDER-MIB .1.3.6.1

# Forcer une redécouverte avec les nouvelles MIBs
sudo -u librenms php discovery.php -h 192.168.10.20 -m mib-discovery
```

5. Alertes et Notifications

Le moteur d'alertes de LibreNMS est l'un de ses points forts. Il permet de définir des règles complexes basées sur n'importe quelle métrique collectée, avec des conditions AND/OR imbriquées, des seuils dynamiques et des templates de notification entièrement personnalisables.

5.1 Concepts fondamentaux du moteur d'alertes

Le moteur d'alertes LibreNMS fonctionne selon les principes suivants :

Rules (Règles) : Définissent les conditions déclenchant une alerte (ex: utilisation CPU > 90% pendant 5 minutes)

Templates : Définissent le format du message d'alerte (sujet, corps, variables)

Transports : Définissent comment envoyer l'alerte (email, Slack, PagerDuty, webhook...)

Alert Groups : Regroupent les devices pour cibler les alertes

Escalation : Remontée automatique si l'alerte n'est pas acquittée

5.2 Règles d'alerte essentielles

Voici les règles d'alerte fondamentales à configurer dans tout déploiement LibreNMS :

Règle	Condition	Sévérité	Délai
Device Down	devices.status = 0	Critical	0 min
CPU > 90%	processors.processor_usage > 90	Warning	5 min
CPU > 95%	processors.processor_usage > 95	Critical	2 min
Mémoire > 85%	mempools_perc > 85	Warning	5 min
Disque > 80%	storage_perc > 80	Warning	10 min
Interface Down	ports.ifOperStatus = "down"	Critical	0 min
Erreurs CRC	port_in_crc > 100/min	Warning	5 min
BGP Peer Down	bgpPeers.bgpPeerState != "established"	Critical	0 min

5.3 Création de règles via l'interface web

```
 # Accéder à l'interface d'alertes
# Menu : Alerts > Alert Rules > + Add Rule

# Exemple de règle pour CPU élevé
# Name: CPU Critical > 95%
# Builder mode: Advanced (SQL)
# Rule:
macros.device = 1
AND processors.processor_usage > 95

# Ou en mode "Query Builder" :
# Table: Processors
# Condition: processor_usage > 95
# AND macros.device = 1

# Paramètres de la règle :
# Severity: critical
# Delay: 120 (secondes avant première alerte)
# Interval: 300 (secondes entre répétitions)
# Count: 3 (nombre de fois avant alerte)
# Mute: false
```

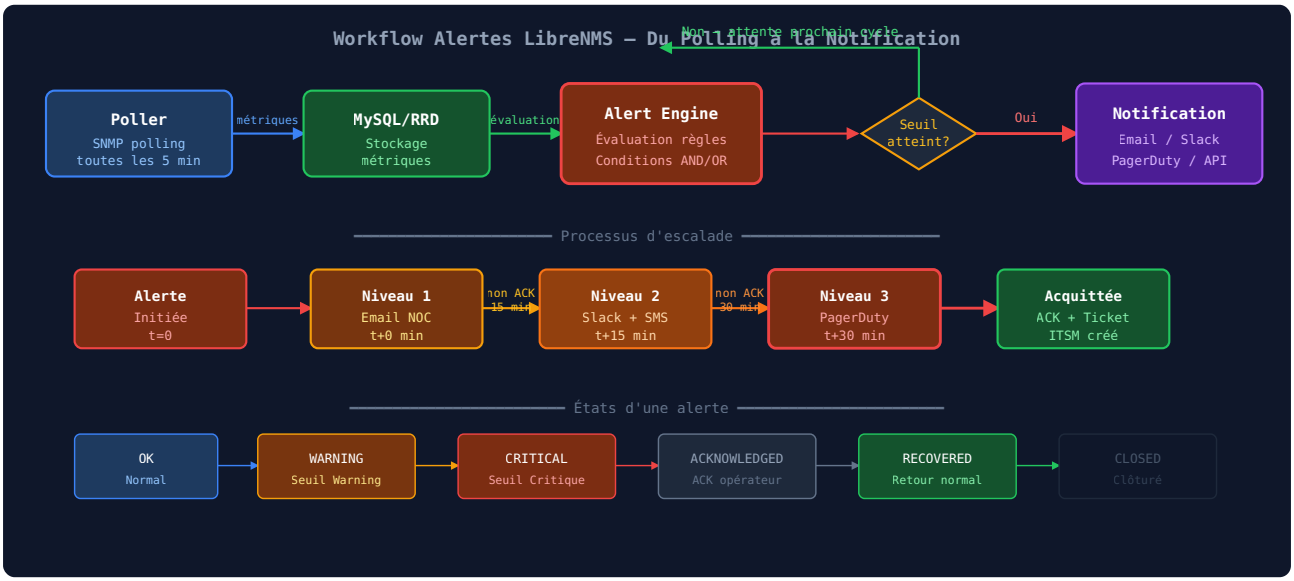
5.4 Templates d'alertes personnalisés

```
 # Template d'alerte Jinja2 pour LibreNMS
# Menu: Alerts > Alert Templates > + Add Template

# Sujet (Subject):
[LibreNMS] {{ $alert->severity | upper }} - {{ $alert->name }} sur {{ $alert->hostname }}

# Corps (Body HTML) :
<h2>Alerte LibreNMS — {{ $alert->severity | upper }}</h2>
<table>
<tr><td><strong>Équipement</strong></td><td>{{ $alert->hostname }}</td></tr>
<tr><td><strong>IP</strong></td><td>{{ $alert->sysName }}</td></tr>
<tr><td><strong>Règle</strong></td><td>{{ $alert->name }}</td></tr>
<tr><td><strong>Sévérité</strong></td><td>{{ $alert->severity }}</td></tr>
<tr><td><strong>Heure</strong></td><td>{{ $alert->timestamp }}</td></tr>
<tr><td><strong>Détail</strong></td><td>{{ $alert->details }}</td></tr>
</table>
<p><a href="{{ $alert->url }}">Voir dans LibreNMS</a></p>

@foreach ($alert->faults as $key => $value)
<p>{{ $value['string'] }}</p>
@endforeach
```

5.5 Transport Slack

```
 # Configuration du transport Slack
# Menu: Alerts > Alert Transports > + Add Transport
# Type: Slack

# Paramètres :
# Name: Slack-NOC
# Webhook URL: https://hooks.slack.com/services/TXXXXX/BXXXXX/XXXXX
# Channel: #noc-alerts
# Username: LibreNMS-Bot
# Icon: :warning:

# Pour PagerDuty (intégration PD Events API v2) :
# Type: PagerDuty
# Integration Key: xxxxxxxxxxxxxxxxxxx (récupéré depuis PagerDuty Service)
# Routing Key: (optionnel, pour routing)

# Pour Microsoft Teams :
# Type: Microsoft Teams
# Webhook URL: https://outlook.office.com/webhook/xxx/IncomingWebhook/xxx

# Exemple de configuration API (via curl)
curl -X POST "https://librenms.votre-domaine.fr/api/v0/alert_transports" \
  -H "X-Auth-Token: VotreAPIToken" \
  -H "Content-Type: application/json" \
  -d '{
    "transport_name": "Slack-NOC",
    "transport_type": "slack",
    "is_default": true,
    "transport_config": {
      "url": "https://hooks.slack.com/services/T.../B.../xxx",
      "channel": "#noc-alerts",
      "username": "LibreNMS",
      "icon": ":warning:"
    }
  }'
```

5.6 Workflow d'alerte — Diagramme SVG

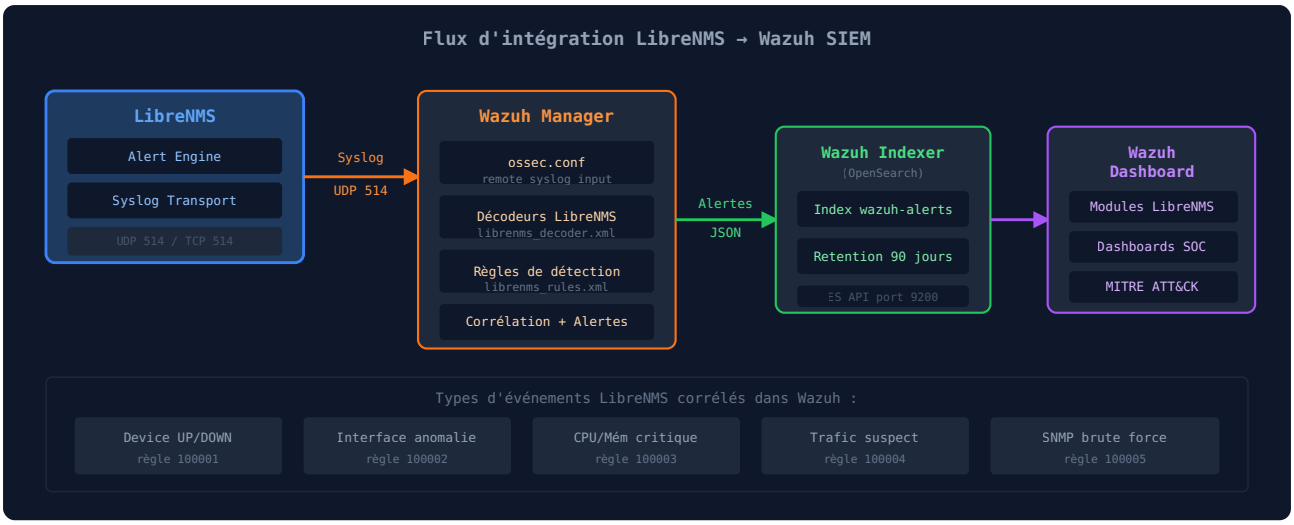


6. Intégration avec Wazuh SIEM

Wazuh est un SIEM/XDR open source leader du marché. L'intégration avec LibreNMS permet de corréler les événements réseau (alertes SNMP, changements de topologie, anomalies d'interfaces) avec les logs d'endpoints, les alertes IDS et les événements de sécurité pour une vision holistique de la sécurité de l'infrastructure.

6.1 Architecture d'intégration LibreNMS → Wazuh

L'intégration repose sur le protocole syslog. LibreNMS envoie ses alertes au format syslog vers le serveur Wazuh (port UDP/TCP 514 par défaut ou un port personnalisé). Wazuh traite ces logs via son moteur de règles et génère des alertes corrélées.



6.2 Configuration du transport syslog dans LibreNMS

```
 # Dans LibreNMS : Menu Alerts > Alert Transports > Add Transport
# Type: API (utilisation de l'API Wazuh) ou Syslog

# Option 1 : Transport syslog natif
# Type: Syslog
# Host: 192.168.100.10 (IP Wazuh Manager)
# Port: 514
# Facility: daemon
# Level: critical

# Option 2 : Via le fichier de configuration LibreNMS
# /opt/librenms/config/config.php
$config['alert']['transports']['syslog']['host'] = '192.168.100.10';
$config['alert']['transports']['syslog']['port'] = 514;
$config['alert']['transports']['syslog']['facility'] = LOG_DAEMON;

# Option 3 : rsyslog sur le serveur LibreNMS pour forwarding
cat >> /etc/rsyslog.d/50-librenms-wazuh.conf << 'EOF'
# Forward LibreNMS logs vers Wazuh Manager
:programname, isequal, "librenms" @@192.168.100.10:514
:programname, isequal, "librenms-poller" @@192.168.100.10:514
EOF

systemctl restart rsyslog
```

6.3 Configuration de Wazuh pour recevoir les logs LibreNMS

```
 # /var/ossec/etc/ossec.conf (section remote)
# Ajouter un remote syslog listener

<ossec_config>
  <remote>
    <connection>syslog</connection>
    <port>514</port>
    <protocol>udp</protocol>
    <allowed-ips>192.168.10.50</allowed-ips>  <!-- IP LibreNMS -->
    <local_ip>192.168.100.10</local_ip>
  </remote>
</ossec_config>

# Redémarrer Wazuh Manager
systemctl restart wazuh-manager

# Vérifier que le port est ouvert
ss -tulnp | grep 514
```

6.4 Décodeur Wazuh personnalisé pour LibreNMS

```
 # Fichier: /var/ossec/etc/decoders/librenms_decoder.xml

<decoder name="librenms">
  <prematch>LibreNMS Alert|librenms|LIBRENMS</prematch>
</decoder>

<decoder name="librenms_alert">
  <parent>librenms</parent>
  <regex>CRITICAL: (\S+) - (\S+) (\w+) (\S+)</regex>
  <order>hostname, rule_name, status, metric_value</order>
</decoder>

<decoder name="librenms_device_down">
  <parent>librenms</parent>
  <regex>Device (\S+) \((\S+)\) is (DOWN|UP)</regex>
  <order>hostname, ip, status</order>
</decoder>

<decoder name="librenms_interface">
  <parent>librenms</parent>
  <regex>Interface (\S+) on (\S+) is (down|up)</regex>
  <order>interface, hostname, status</order>
</decoder>
```

6.5 Règles Wazuh pour alertes LibreNMS

```
 # Fichier: /var/ossec/etc/rules/librenms_rules.xml

<group name="librenms,network">

  <!-- Règle de base LibreNMS -->
  <rule id="100001" level="5">
    <decoded_as>librenms</decoded_as>
    <description>LibreNMS: Événement réseau détecté</description>
    <group>librenms</group>
  </rule>

  <!-- Device DOWN —  niveau critique -->
  <rule id="100002" level="12">
    <if_sid>100001</if_sid>
    <decoded_as>librenms_device_down</decoded_as>
    <field name="status">DOWN</field>
    <description>LibreNMS: Équipement réseau hors ligne - $(hostname)</description>
    <mitre>
      <id>T1499</id>  <!-- Endpoint Denial of Service -->
    </mitre>
    <group>librenms,network_down</group>
  </rule>

  <!-- Interface DOWN -->
  <rule id="100003" level="8">
    <if_sid>100001</if_sid>
    <decoded_as>librenms_interface</decoded_as>
    <field name="status">down</field>
    <description>LibreNMS: Interface réseau down - $(interface) sur $(hostname)</description>
    <group>librenms,interface_down</group>
  </rule>

  <!-- CRITIQUE: Multiples devices down (possible attaque DDoS/coupure réseau) -->
  <rule id="100004" level="15" frequency="5" timeframe="300">
    <if_matched_sid>100002</if_matched_sid>
    <description>LibreNMS: Multiples équipements hors ligne en 5 minutes (incident réseau majeur)
    <mitre>
      <id>T1498</id>  <!-- Network Denial of Service -->
    </mitre>
    <group>librenms,network_incident</group>
  </rule>

  <!-- CPU critique -->
  <rule id="100005" level="9">
    <if_sid>100001</if_sid>
    <match>CPU Critical</match>
    <description>LibreNMS: Utilisation CPU critique sur $(hostname)</description>
    <group>librenms,performance</group>
  </rule>

</group>
```

```
# Recharger les règles Wazuh
/var/ossec/bin/wazuh-logtest
systemctl restart wazuh-manager
```

6.6 Corrélation avancée : LibreNMS + authentifications

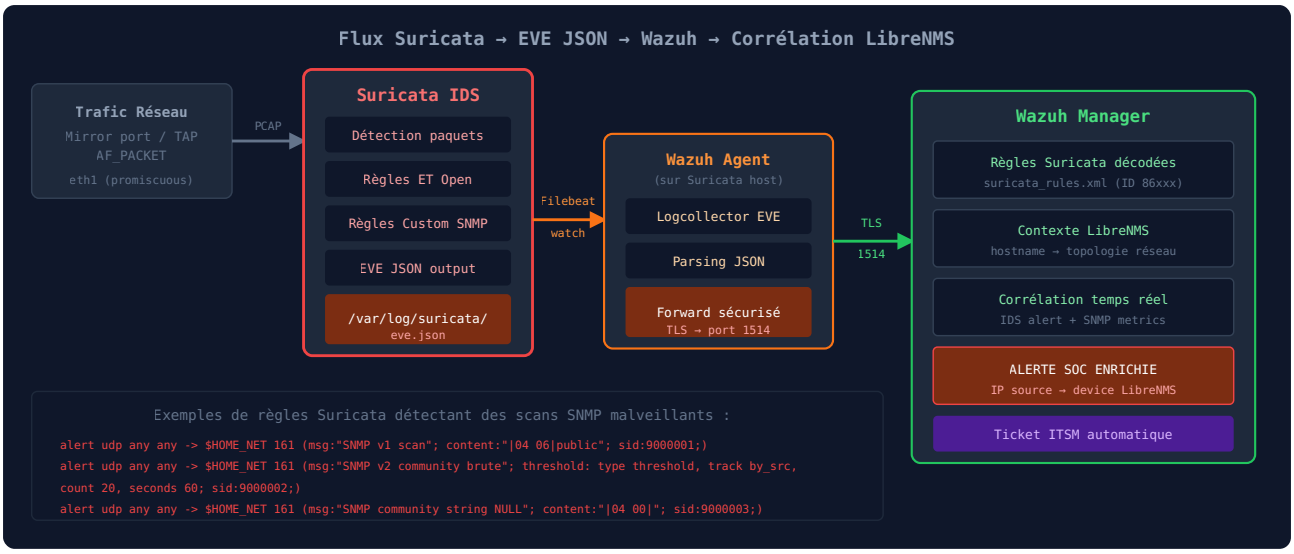
```
 # Règle de corrélation : Device Down + tentative d'authentification
# (scénario : attaque sur équipement réseau → crash → tentative d'accès physique)
<rule id="100010" level="14">
  <if_sid>100002</if_sid>  <!-- Device DOWN LibreNMS -->
  <if_matched_sid>5710</if_matched_sid>  <!-- SSH brute force Wazuh -->
  <timeframe>600</timeframe>
  <description>Possible attaque corrélée: équipement réseau down + tentative SSH sur même sous-ré
  <mitre>
    <id>T1110</id>  <!-- Brute Force -->
    <id>T1498</id>  <!-- Network DoS -->
  </mitre>
</rule>
```

7. Intégration avec Suricata IDS

Suricata est le moteur IDS/IPS open source de référence, maintenu par l'OISF (Open Information Security Foundation). Son intégration avec LibreNMS permet de corréler la topologie réseau supervisée avec les alertes de détection d'intrusion, offrant un contexte riche pour l'investigation des incidents.

7.1 Architecture Suricata + LibreNMS

Dans notre architecture, Suricata surveille le trafic réseau en mode IDS (sans blocage, sur une interface mirror/TAP), tandis que LibreNMS surveille l'état et les métriques des équipements. Les deux systèmes alimentent Wazuh qui assure la corrélation.


Flux Suricata → EVE JSON → Wazuh → Corrélation LibreNMS
Trafic Réseau
Mirror port / TAP
AF_PACKET
eth1 (promiscuous)
PCAP
Suricata IDS
Détection paquets
Règles ET Open
Règles Custom SNMP
EVE JSON output
/var/log/suricata/
eve.json
Filebeat
watch
Wazuh Agent
(sur Suricata host)
Logcollector EVE
Parsing JSON
Forward sécurisé
TLS → port 1514
TLS
1514
Wazuh Manager
Règles Suricata décodées
suricata_rules.xml (ID 86xxx)
Contexte LibreNMS
hostname → topologie réseau
Corrélation temps réel
IDS alert + SNMP metrics
ALERTE SOC ENRICHIE
IP source → device LibreNMS
Ticket ITSM automatique
Exemples de règles Suricata détectant des scans SNMP malveillants :
alert udp any any -> $HOME_NET 161 (msg:"SNMP v1 scan"; content:"|04 06|public"; sid:9000001;)
alert udp any any -> $HOME_NET 161 (msg:"SNMP v2 community brute"; threshold: type threshold, track by_src,
count 20, seconds 60; sid:9000002;)
alert udp any any -> $HOME_NET 161 (msg:"SNMP community string NULL"; content:"|04 00|"; sid:9000003;)

7.2 Installation et configuration de Suricata

```
 # Installation Suricata sur Ubuntu 22.04
add-apt-repository ppa:oisf/suricata-stable -y
apt update && apt install suricata suricata-update -y

# Mise à jour des règles Emergent Threats (ET Open)
suricata-update
suricata-update list-sources
suricata-update enable-source et/open  # Gratuit
# suricata-update enable-source et/pro  # Payant, meilleure couverture

# Configuration principale (/etc/suricata/suricata.yaml)
cat > /etc/suricata/suricata.yaml << 'SURICONF'
%YAML 1.1
---
# Réseau local (adapter à votre infrastructure)
vars:
  address-groups:
    HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
    EXTERNAL_NET: "!$HOME_NET"
    HTTP_SERVERS: "$HOME_NET"
    DNS_SERVERS: "$HOME_NET"
    SMTP_SERVERS: "$HOME_NET"

# Interface de capture (mirror port du switch)
af-packet:
  - interface: eth1
    cluster-id: 99
    cluster-type: cluster_flow
    defrag: yes
    use-mmap: yes
    tpacket-v3: yes
    ring-size: 200000
    block-size: 32768
    block-timeout: 10
    use-emergency-flush: yes

# Sortie EVE JSON (pour Wazuh/Filebeat)
outputs:
  - eve-log:
      enabled: yes
      filetype: regular
      filename: /var/log/suricata/eve.json
      community-id: true  # Pour corrélation avec Zeek/NetworkMiner
      types:
        - alert:
            payload: yes
            payload-buffer-size: 4kb
            http-body: yes
            http-body-printable: yes
            metadata: yes
            tagged-packets: yes
```

```
        - http:
            extended: yes
        - dns:
        - tls:
            extended: yes
        - files:
            force-magic: no
        - smtp:
        - flow
        - netflow
SURICONF

# Démarrage
systemctl enable suricata
systemctl start suricata
```

7.3 Règles Suricata pour détecter les scans SNMP

```
 # Fichier: /etc/suricata/rules/snmp-monitoring.rules

# Détection de scan SNMP v1/v2c avec community "public"
alert udp any any -> $HOME_NET 161 (
    msg:"ET SCAN SNMP v1 community public (scan potentiel)";
    content:"|04 06|public";
    threshold: type threshold, track by_src, count 10, seconds 60;
    classtype:attempted-recon;
    sid:9100001; rev:1;)

# Community string vide (attaque)
alert udp any any -> $HOME_NET 161 (
    msg:"ET SCAN SNMP community string vide";
    content:"|04 00|";
    classtype:policy-violation;
    sid:9100002; rev:1;)

# Scan de masse SNMP (plus de 100 requêtes/minute depuis une source)
alert udp any any -> $HOME_NET 161 (
    msg:"ET SCAN SNMP bulk scan depuis IP unique";
    threshold: type both, track by_src, count 100, seconds 60;
    classtype:attempted-recon;
    sid:9100003; rev:1;)

# Tentative de GetBulk (enumération de masse)
alert udp any any -> $HOME_NET 161 (
    msg:"ET SCAN SNMP GetBulk Request (enumération)";
    content:"|a5|";  # GetBulkRequest PDU type
    classtype:attempted-recon;
    sid:9100004; rev:1;)

# SNMP SET (tentative de modification)
alert udp any any -> $HOME_NET 161 (
    msg:"ET POLICY SNMP SET Request (modification config)";
    content:"|a3|";  # SetRequest PDU type
    classtype:policy-violation;
    sid:9100005; rev:1;)

# Vérifier la règle
suricata-update --no-merge
suricata -T -c /etc/suricata/suricata.yaml
```

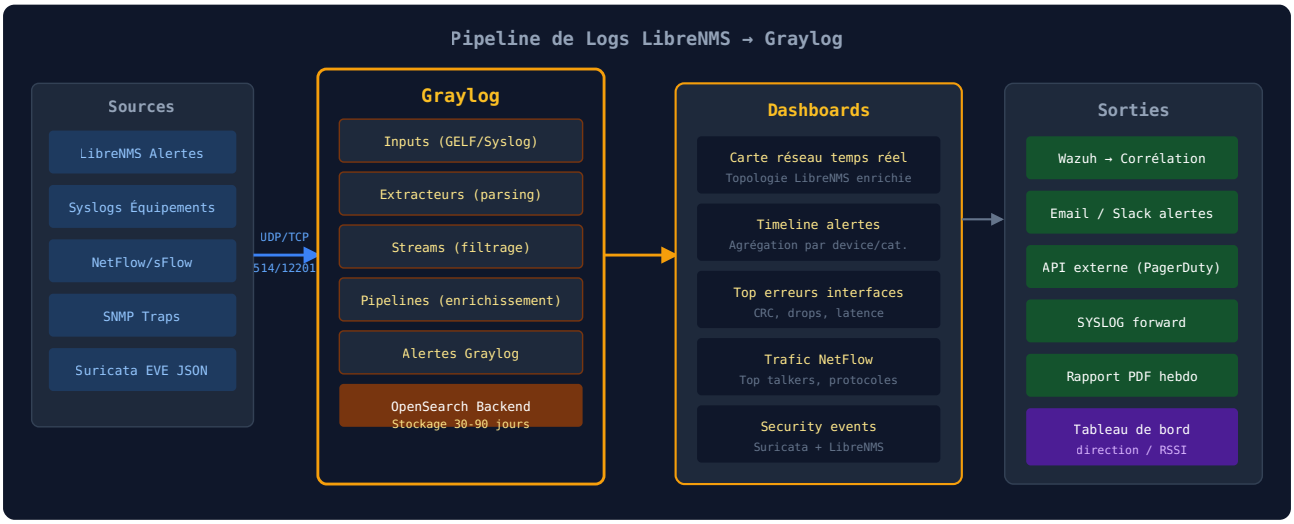
7.4 Configuration de l'agent Wazuh pour Suricata

```
 # /var/ossec/etc/ossec.conf sur le host Suricata
<ossec_config>
  <localfile>
    <log_format>json</log_format>
    <location>/var/log/suricata/eve.json</location>
    <label key="@source">suricata</label>
  </localfile>
</ossec_config>

# Règles Wazuh pour Suricata (déjà incluses dans Wazuh 4.x)
# /var/ossec/ruleset/rules/0800-suricata_rules.xml
# Mais ajoutons des règles de corrélation LibreNMS + Suricata

# /var/ossec/etc/rules/suricata_librenms_correlation.xml
<group name="suricata,librenms,correlation">

  <!-- Alerte Suricata sur un device surveillé par LibreNMS -->
  <rule id="200001" level="12">
    <if_sid>86601</if_sid>  <!-- Suricata alert -->
    <if_matched_sid>100001</if_matched_sid>  <!-- LibreNMS event -->
    <timeframe>3600</timeframe>
    <description>Corrélation: alerte IDS Suricata ET événement réseau LibreNMS sur même hôte</des
    <mitre>
      <id>T1046</id>  <!-- Network Service Scanning -->
    </mitre>
  </rule>

  <!-- Scan SNMP détecté → Device LibreNMS ciblé -->
  <rule id="200002" level="14">
    <if_sid>86601</if_sid>
    <match>SNMP scan|SNMP community|SNMP bulk</match>
    <description>Suricata: Scan SNMP détecté, tentative d'énumération d'équipements supervisés</d
    <mitre>
      <id>T1595</id>  <!-- Active Scanning -->
    </mitre>
  </rule>

</group>
```

8. Intégration avec Graylog

Graylog est une plateforme de gestion centralisée des logs (SIEM léger) très utilisée avec LibreNMS pour la centralisation des syslogs réseau, l'analyse des logs de supervision et la création de dashboards opérationnels. Contrairement à Wazuh (orienté sécurité), Graylog excelle dans la collecte massive de logs, l'enrichissement et la visualisation opérationnelle.

8.1 Architecture d'intégration Graylog



8.2 Configuration du transport GELF dans LibreNMS

```
 # LibreNMS → Graylog via GELF (Graylog Extended Log Format)
# GELF est préféré à syslog car il supporte les champs structurés JSON

# Dans LibreNMS, configurer le transport API vers Graylog
# Alerts > Alert Transports > Add Transport
# Type: API

# Configuration via la base LibreNMS (ou interface web)
# Webhook vers l'API Graylog GELF HTTP Input

# Ou via rsyslog avec format GELF
cat > /etc/rsyslog.d/60-graylog.conf << 'EOF'
# Template GELF pour Graylog
template(name="gelf" type="list") {
    constant(value="{")
    constant(value=""version":"1.1",")
    constant(value=""host":"")
    property(name="hostname")
    constant(value="",")
    constant(value=""short_message":"")
    property(name="msg" format="json")
    constant(value="",")
    constant(value=""timestamp":")
    property(name="timegenerated" dateformat="unixtimestamp")
    constant(value=",")
    constant(value=""level":")
    property(name="syslogseverity")
    constant(value=",")
    constant(value=""_facility":"")
    property(name="syslogfacility-text")
    constant(value="",")
    constant(value=""_source":"librenms"")
    constant(value="}")
}

# Envoi vers Graylog GELF UDP Input (port 12201)
:programname, isequal, "librenms"   @192.168.100.20:12201;gelf
:programname, isequal, "rsyslogd"   stop

EOF
systemctl restart rsyslog
```

8.3 Configuration de l'Input Graylog pour LibreNMS

```
 # Graylog : System > Inputs > Launch new Input

# Input 1 : GELF UDP (pour LibreNMS via rsyslog)
Type: GELF UDP
Title: LibreNMS-GELF
Bind address: 0.0.0.0
Port: 12201
Buffer size: 1048576

# Input 2 : Syslog UDP (pour syslogs directs des équipements)
Type: Syslog UDP
Title: Network-Syslog
Bind address: 0.0.0.0
Port: 514
Force RDNS: false
Store full message: true
Allow override date: true

# Input 3 : Raw/Plaintext TCP (pour Suricata EVE JSON via Filebeat)
Type: Beats
Title: Suricata-Beats
Bind address: 0.0.0.0
Port: 5044
TLS enabled: true

# Vérifier les inputs actifs via l'API Graylog
curl -u admin:VotrePassword -H "X-Requested-By: api" \
    http://localhost:9000/api/system/inputs | python3 -m json.tool
```

8.4 Extracteurs Graylog pour les alertes LibreNMS

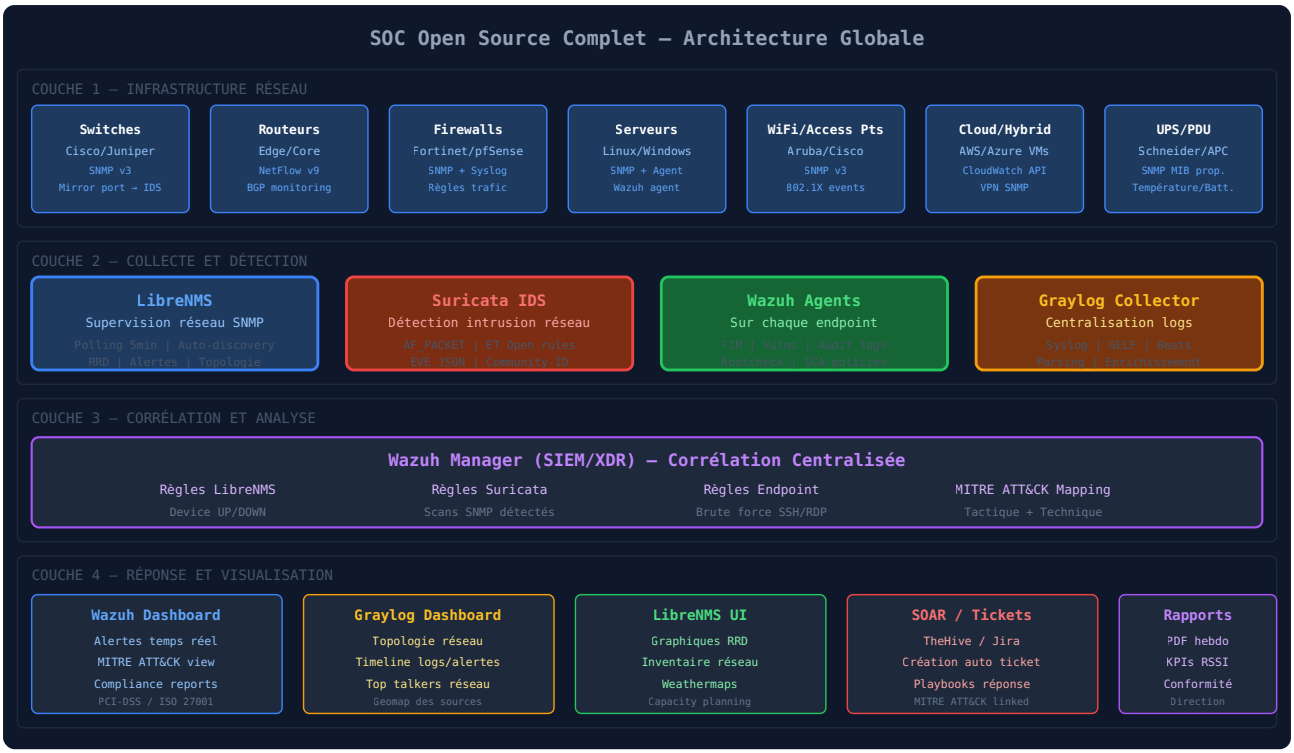
```
 # Extracteur Regex pour parser les alertes LibreNMS syslog
# Graylog : System > Inputs > [Votre Input] > Manage Extractors

# Extracteur 1 : Hostname LibreNMS
Type: Regex
Source field: message
Regex: Device\s+(\S+)\s+is\s+(UP|DOWN)
Target field: librenms_device
Condition: string contains "LibreNMS"

# Extracteur 2 : Sévérité alerte
Type: Regex
Source field: message
Regex: (CRITICAL|WARNING|OK):\s+(.+)
Target field: alert_severity, alert_message

# Extracteur 3 : IP source dans les alertes Suricata
Type: JSON
Source field: message
Key prefix: suricata_
List separator: ,

# Via l'API Graylog (plus pratique pour le déploiement en masse)
curl -X POST "http://localhost:9000/api/system/inputs/{input_id}/extractors" \
  -u admin:VotrePassword \
  -H "Content-Type: application/json" \
  -H "X-Requested-By: api" \
  -d '{
    "title": "LibreNMS Device Status",
    "type": "regex",
    "source_field": "message",
    "target_field": "librenms_device_status",
    "extractor_config": {
      "regex_value": "Device\s+(\S+)\s+is\s+(UP|DOWN)"
    },
    "converters": [],
    "condition_type": "string",
    "condition_value": "LibreNMS"
  }'
```

8.5 Streams et Pipelines Graylog

```
 # Stream 1: Toutes les alertes LibreNMS
Nom: LibreNMS-Alerts
Règle: source matches regex "librenms|LIBRENMS"
Index Set: librenms-alerts (retention 90 jours)

# Stream 2: Alertes critiques uniquement
Nom: LibreNMS-Critical
Règle: alert_severity = "CRITICAL"
Index Set: librenms-critical (retention 1 an)
Notification: Email + Slack immédiat

# Stream 3: Sécurité réseau (Suricata + LibreNMS)
Nom: Network-Security
Règle: source matches "suricata" OR source matches "librenms"
Index Set: network-security (retention 1 an)

# Pipeline pour enrichissement avec GeoIP
# (identifier la géolocalisation des IPs sources d'attaque)
rule "enrich_source_ip"
when
  has_field("src_ip")
then
  let geo = lookup("geoip", to_string($message.src_ip));
  set_field("src_country", geo["country"]);
  set_field("src_city", geo["city"]);
  set_field("src_latitude", geo["latitude"]);
  set_field("src_longitude", geo["longitude"]);
end
```

9. Stack SOC Complète : LibreNMS + Wazuh + Suricata + Graylog

L'intégration de ces quatre outils open source forme un SOC (Security Operations Center) complet et performant, capable de couvrir les besoins de supervision réseau, détection d'intrusion, corrélation SIEM et visualisation opérationnelle d'une PME ou d'un département IT d'entreprise.



9.1 Cas d'usage SOC : Détection d'un scan réseau

Voici le flux complet d'un incident typique : un attaquant effectue un scan réseau depuis Internet qui atteint le VLAN de management :

Étape	Outil	Événement	Action
T+0	Suricata	Scan SNMP UDP 161 depuis 203.x.x.x	Alerte EVE JSON générée
T+1s	Wazuh Agent	Lecture eve.json, forward au Manager	Transmission TLS port 1514
T+2s	Wazuh Manager	Règle 200002 déclenchée (SNMP scan)	Alerte level 14 créée
T+5s	LibreNMS	Latence anormale sur switch core	Alerte syslog → Wazuh
T+7s	Wazuh Manager	Corrélation : scan + anomalie SNMP	Alerte critique level 15
T+10s	Graylog	Log indexé, dashboard mis à jour	Géolocalisation IP source
T+15s	PagerDuty	Appel téléphonique astreinte NOC	Incident ouvert
T+30s	Firewall	Blocage IP 203.x.x.x via API	Réponse automatique SOAR

10. Sécurisation de LibreNMS

LibreNMS est un composant critique de votre infrastructure de sécurité. Sa compromission permettrait à un attaquant de cartographier l'ensemble de votre réseau, d'accéder aux credentials SNMP, voire de modifier la configuration des équipements supervisés. Sa sécurisation est donc primordiale.

10.1 Durcissement Nginx (TLS 1.3, HSTS, CSP)

```
 # Configuration Nginx sécurisée pour LibreNMS
server {
    listen 443 ssl http2;
    server_name librenms.votre-domaine.fr;
    ssl_certificate /etc/letsencrypt/live/librenms.votre-domaine.fr/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/librenms.votre-domaine.fr/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-S
    ssl_prefer_server_ciphers off;
    ssl_session_timeout 1d;
    ssl_session_cache shared:SSL:50m;
    add_header Strict-Transport-Security "max-age=63072000; includeSubDomains; preload" always;
    add_header X-Frame-Options "DENY" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header Referrer-Policy "strict-origin-when-cross-origin" always;
    add_header Content-Security-Policy "default-src 'self'; script-src 'self' 'unsafe-inline' 'uns
    ssl_stapling on;
    ssl_stapling_verify on;
    resolver 1.1.1.1 8.8.8.8 valid=300s;
    client_max_body_size 10M;
    limit_req_zone $binary_remote_addr zone=librenms_login:10m rate=5r/m;
    location /login {
        limit_req zone=librenms_login burst=10 nodelay;
        try_files $uri $uri/ /index.php?$query_string;
    }
    root /opt/librenms/html;
    index index.php;
    location ~* /(\.git|\.env|composer\.(json|lock)|package\.json) {
        deny all; return 404;
    }
    location ~* \.(ini|conf|yaml|yml|bak|old)$ { deny all; }
}
```

10.2 Authentification LDAP / Active Directory

```
 # /opt/librenms/config/config.php
$config['auth_mechanism'] = "ldap";
$config['auth_ldap_server'] = "dc01.votre-domaine.fr";
$config['auth_ldap_port'] = 636;  // LDAPS
$config['auth_ldap_use_tls'] = 1;
$config['auth_ldap_version'] = 3;
$config['auth_ldap_suffix'] = ",dc=votre-domaine,dc=fr";
$config['auth_ldap_groupbase'] = "ou=Groups,dc=votre-domaine,dc=fr";
$config['auth_ldap_userbase'] = "ou=Users,dc=votre-domaine,dc=fr";
$config['auth_ldap_binddn'] = "cn=svc-librenms,ou=ServiceAccounts,dc=votre-domaine,dc=fr";
$config['auth_ldap_bindpassword'] = "MotDePasseServiceAccount2026!";
$config['auth_ldap_uid_attribute'] = "sAMAccountName";
$config['auth_ldap_groups']['NOC-Team']['level'] = 5;      // Utilisateur standard
$config['auth_ldap_groups']['Network-Admin']['level'] = 9; // Admin complet
$config['auth_ldap_groups']['RSSI']['level'] = 10;         // Super admin
$config['auth_ldap_filter'] = "(memberOf=cn=LibreNMS-Users,ou=Groups,dc=votre-domaine,dc=fr)";
```

10.3 Isolation réseau via VLAN de management

```
 # Règles iptables pour isoler LibreNMS
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -s 192.168.100.0/24 -p tcp --dport 22 -j ACCEPT   # SSH admin
iptables -A INPUT -s 192.168.100.0/24 -p tcp --dport 443 -j ACCEPT  # HTTPS
iptables -A INPUT -s 192.168.10.0/24 -p udp --dport 514 -j ACCEPT   # Syslog
iptables -A INPUT -s 192.168.10.0/24 -p tcp --dport 514 -j ACCEPT
iptables -A INPUT -s 192.168.10.0/24 -p udp --dport 162 -j ACCEPT   # SNMP traps
# Bloquer SNMP depuis Internet (obligatoire)
iptables -A INPUT -p udp --dport 161 -j DROP
iptables -A INPUT -p udp --dport 162 -j DROP
iptables-save > /etc/iptables/rules.v4
```

10.4 Audit des versions SNMP utilisées

```
 #!/bin/bash
# Script d'audit : détecter les devices utilisant SNMP v1/v2c (non chiffré)
# /opt/librenms/scripts/audit-snmp-version.sh

MYSQL_CMD="mysql -u librenms -pVotreMotDePasseSecurise2026! -D librenms -N -e"

echo "=== Audit SNMP Version — LibreNMS — $(date) ==="
echo ""
echo "Devices utilisant SNMP v1 (NON SÉCURISÉ) :"
$MYSQL_CMD "SELECT hostname, ip, snmpver FROM devices WHERE snmpver='v1' ORDER BY hostname;"

echo ""
echo "Devices utilisant SNMP v2c (community string en clair) :"
$MYSQL_CMD "SELECT hostname, ip, community, snmpver FROM devices WHERE snmpver='v2c' ORDER BY hos

echo ""
echo "Résumé par version :"
$MYSQL_CMD "SELECT snmpver, COUNT(*) as count FROM devices GROUP BY snmpver;"

V1_COUNT=$($MYSQL_CMD "SELECT COUNT(*) FROM devices WHERE snmpver='v1'")
V2C_COUNT=$($MYSQL_CMD "SELECT COUNT(*) FROM devices WHERE snmpver='v2c'")

if [ "$(($V1_COUNT + $V2C_COUNT))" -gt 0 ]; then
    echo "ALERTE: $(($V1_COUNT + $V2C_COUNT)) device(s) utilisent SNMP non chiffré !"
    echo "Audit SNMP: devices non sécurisés" | mail -s "[LibreNMS] ALERTE SNMP" noc@votre-entrepr
fi
```

10.5 Sauvegarde configuration et données RRD

```
 #!/bin/bash
# Script de sauvegarde LibreNMS — /opt/librenms/scripts/backup-librenms.sh
BACKUP_DIR="/opt/backups/librenms"
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_PATH="$BACKUP_DIR/$DATE"
mkdir -p "$BACKUP_PATH"

# 1. Backup MySQL
mysqldump -u librenms -pVotreMotDePasseSecurise2026! librenms | gzip > "$BACKUP_PATH/librenms_db.s

# 2. Backup configuration
tar -czf "$BACKUP_PATH/librenms_config.tar.gz" \
    /opt/librenms/.env /opt/librenms/config/ \
    /etc/nginx/sites-available/librenms.conf

# 3. Backup RRD data (métriques historiques)
tar -czf "$BACKUP_PATH/librenms_rrd.tar.gz" /opt/librenms/rrd/

# 4. Checksum intégrité
md5sum "$BACKUP_PATH"/* > "$BACKUP_PATH/checksums.md5"

# 5. Rotation 30 jours
find "$BACKUP_DIR" -maxdepth 1 -type d -mtime +30 -exec rm -rf {} \;

echo "Backup terminé: $BACKUP_PATH ($(du -sh $BACKUP_PATH | cut -f1))"
# Cron: 0 2 * * * /opt/librenms/scripts/backup-librenms.sh >> /var/log/librenms-backup.log
```

10.6 Mise à jour automatique via daily.sh

```
 # LibreNMS fournit un script de mise à jour automatique
# Le cron librenms exécute daily.sh toutes les 6h (par défaut)
# 33 */6 * * * /opt/librenms/cronic /opt/librenms/daily.sh

# daily.sh effectue :
# 1. git pull (mise à jour du code source)
# 2. composer install --no-dev (dépendances PHP)
# 3. php artisan migrate (migrations DB)
# 4. Nettoyage logs et données RRD obsolètes

# Pour forcer une mise à jour immédiate
su -s /bin/bash librenms -c "/opt/librenms/daily.sh"

# Vérifier les logs de mise à jour
tail -100 /opt/librenms/logs/librenms.log | grep -i "update\|error\|warning"

# Vérifier la version installée
su -s /bin/bash librenms -c "cd /opt/librenms && php validate.php | head -20"
```

---

11. Performance et Scalabilité

Pour les grandes infrastructures (500+ devices, 10 000+ interfaces), LibreNMS propose une architecture distribuée qui répartit la charge de polling sur plusieurs serveurs. Cette section couvre le distributed polling, l'optimisation des performances et les outils complémentaires.

11.1 Distributed Polling

Le distributed polling permet de déployer plusieurs pollers dans différentes localisations géographiques, tous remontant vers une base de données centrale :

```
 # Architecture Distributed Polling
# Serveur central (DB + Web UI)
# ├── Poller Paris (VLAN 10 - 200 devices)
# ├── Poller Lyon (VLAN 20 - 150 devices)
# └── Poller Bordeaux (VLAN 30 - 100 devices)

# Configuration serveur central (/opt/librenms/config/config.php)
$config['distributed_poller'] = true;
$config['distributed_poller_group'] = 0;   // 0 = master
$config['distributed_poller_name'] = 'Paris-Central';

# Sur chaque poller distant
$config['distributed_poller'] = true;
$config['distributed_poller_group'] = 1;   // 1, 2, 3...
$config['distributed_poller_name'] = 'Lyon-Poller';
$config['db_host'] = '192.168.100.50';     // DB centrale
$config['redis_host'] = '192.168.100.50';  // Redis coordination
$config['redis_port'] = 6379;

# Assigner des devices à un groupe de pollers
# Menu: Devices > Edit Device > Poller Group
```

11.2 Optimisation MySQL pour grande charge

```
 # /etc/mysql/mariadb.conf.d/60-librenms-perf.cnf
[mysqld]
innodb_buffer_pool_size = 4G           # 70-80% de la RAM disponible
innodb_buffer_pool_instances = 8
innodb_log_file_size = 512M
innodb_flush_log_at_trx_commit = 2     # Perf (risque perte ~1s en crash)
innodb_flush_method = O_DIRECT
max_connections = 200
wait_timeout = 28800
table_open_cache = 2000
table_definition_cache = 1000
query_cache_type = 0                   # Désactiver sur MySQL 8+
event_scheduler = ON                   # Pour maintenance auto
```

11.3 Oxidized — Backup automatique des configurations réseau

Oxidized est un outil de sauvegarde des configurations réseau qui s'intègre nativement avec LibreNMS pour versionner automatiquement les configurations de vos équipements dans Git :

```
 # Installation d'Oxidized
gem install oxidized oxidized-script oxidized-web

# Configuration (/etc/oxidized/config)
---
username: admin
password: VotreMotDePasse
model: junos
interval: 3600
use_syslog: true
threads: 30
timeout: 20
retries: 3

source:
  default: http
  http:
    url: http://localhost/api/v0/oxidized/configuration
    scheme: http
    headers:
      X-Auth-Token: VotreAPITokenLibreNMS

output:
  default: git
  git:
    user: LibreNMS Backup
    email: librenms@votre-domaine.fr
    repo: "/var/lib/oxidized/oxidized.git"

# Intégration dans LibreNMS config.php
$config['oxidized']['enabled'] = true;
$config['oxidized']['url'] = 'http://localhost:8888';
$config['oxidized']['features']['versioning'] = true;

systemctl enable oxidized && systemctl start oxidized
```

11.4 Weathermaps et Capacity Planning

```
 # Activation plugin Weathermap dans LibreNMS
php artisan plugin:enable weathermap

# Exemple de configuration weathermap (fichier .conf)
WIDTH 1920
HEIGHT 1080
BGCOLOR #0f172a
TITLE Réseau Production

NODE Core-Switch
  LABEL SW-CORE-01
  POSITION 960 540
  ICON /plugins/Weathermap/images/cisco-switch.png

NODE Firewall
  LABEL FW-EDGE-01
  POSITION 960 200

LINK Core-to-FW
  NODES Core-Switch Firewall
  TARGET librenms:10:ge-0/0/0:in librenms:10:ge-0/0/0:out
  BANDWIDTH 10G
  BWLABEL bits

# Script capacity planning — interfaces > 80% utilisation
# /opt/librenms/scripts/capacity-planning.sh
mysql -u librenms -pVotreMotDePasseSecurise2026! librenms -e "
SELECT d.hostname, p.ifDescr, p.ifSpeed/1000000 as speed_mbps,
  GREATEST(
    p.ifInOctets_rate*8/p.ifSpeed*100,
    p.ifOutOctets_rate*8/p.ifSpeed*100
  ) as usage_pct
FROM ports p JOIN devices d ON p.device_id=d.device_id
WHERE p.ifSpeed > 0
  AND GREATEST(p.ifInOctets_rate*8/p.ifSpeed*100, p.ifOutOctets_rate*8/p.ifSpeed*100) > 80
ORDER BY usage_pct DESC LIMIT 20;"
```

11.5 Monitoring de la santé de LibreNMS lui-même

```
 # LibreNMS dispose d'un module de self-monitoring
# Accessible via /overview et l'API

# Métriques à surveiller sur LibreNMS :
# - Temps de polling moyen (doit rester < 300s pour un cycle de 5min)
# - Devices en timeout SNMP (onglet "Devices > Not Responding")
# - Taille de la base de données MySQL
# - Utilisation disque des RRD

# Vérification de la santé du polling
curl -s -H "X-Auth-Token: VotreAPIToken" \
  "https://librenms.votre-domaine.fr/api/v0/system" | python3 -m json.tool

# Vérifier les devices en erreur de polling
curl -s -H "X-Auth-Token: VotreAPIToken" \
  "https://librenms.votre-domaine.fr/api/v0/devices?type=ignored" | python3 -m json.tool

# Alerter si le poller prend plus de 5 minutes
# (indique un sous-dimensionnement du serveur LibreNMS)
su -s /bin/bash librenms -c "cd /opt/librenms && php poll-all.php 2>&1" | \
  grep -i "poll time\|warning\|error"
```

12. FAQ LibreNMS

Comment migrer de SNMP v2c vers SNMP v3 sans interruption de supervision ?

La migration de SNMP v2c vers v3 se fait en deux étapes pour éviter toute interruption. Commencez par configurer SNMP v3 en parallèle sur vos équipements réseau, en gardant temporairement SNMP v2c actif. Une fois SNMP v3 configuré et testé via `snmpget -v3 -l authPriv -u librenmsv3 -a SHA-256 ...`, mettez à jour le device dans LibreNMS : accédez à Devices > Edit > SNMP et changez la version vers v3 avec les paramètres authPriv, SHA-256 et AES-128. Après validation que le polling LibreNMS fonctionne en v3 (vérifiez dans les logs sur 2 cycles de 5 minutes), désactivez SNMP v2c

LibreNMS peut-il superviser des équipements AWS EC2 et Azure VM sans agent SNMP ? +

Comment éviter les faux positifs lors des maintenances planifiées dans LibreNMS ? +

Quelle est la différence entre LibreNMS, Zabbix et Nagios pour un SOC en 2026 ? +

Comment diagnostiquer et résoudre les problèmes de polling SNMP dans LibreNMS ? +

Le diagnostic des problèmes de polling suit une démarche structurée. Commencez par tester SNMP manuellement depuis le serveur LibreNMS vers le device ciblé : `snmpget -v3 -l authPriv -u librenmsv3 -a SHA-256 -A "MotDePasseAuth" -x AES -X "MotDePassePriv" ADRESSE_IP .1.3.6.1.2.1.1.1.0`. Si la commande échoue, le problème est réseau (firewall, ACL, mauvais credentials). Si elle réussit mais LibreNMS ne poll pas, vérifiez les logs LibreNMS : `tail -f /opt/librenms/logs/librenms.log | grep DEVICE_IP`. Forcez un poll manuel avec `sudo -u librenms php /opt/librenms/poll.php -h DEVICE_IP -d` — le mode debug (-d) affiche tous les OIDs interrogés et les erreurs. Les causes communes incluent : credentials

Comment configurer LibreNMS pour envoyer des alertes vers un webhook Webhook.site ou n8n pour l'automatisation ?

À RETENIR

Points clés LibreNMS

LibreNMS est le NEMS open source de référence en 2026 : fork GPLv3 d'Observium, PHP/Laravel, 650+ contributeurs, support de 400+ fabricants réseau

SNMP v3 obligatoire en production : utiliser systématiquement le niveau authPriv avec SHA-256 (authentification) et AES-128 (chiffrement) — bannir v1/v2c

Installation officielle sur Ubuntu 22.04/24.04 : suivre scrupuleusement la documentation officielle, valider avec `php validate.php` avant toute mise en production

Moteur d'alertes avancé : règles SQL complexes, templates Jinja2, transports multiples (email, Slack, PagerDuty, webhook), escalade configurable, fenêtres de maintenance

Intégration Wazuh SIEM : envoyer les logs LibreNMS via syslog UDP 514, créer des décodeurs et règles personnalisés pour corréler les événements réseau avec les alertes de sécurité

Suricata IDS en complément : détecter les scans SNMP malveillants, les brute-forces de community strings, les énumérations réseau — corrélation via Wazuh avec le contexte topologique LibreNMS

Graylog pour la centralisation des logs : GELF/syslog depuis LibreNMS et les équipements réseau, pipelines d'enrichissement, dashboards opérationnels temps réel

Sécurisation LibreNMS : TLS 1.3 + HSTS + CSP sur Nginx, authentification LDAP/AD, isolation VLAN management, iptables restreint, audit SNMP régulier

Scalabilité : Distributed Polling pour 10 000+ devices, Redis pour la coordination, Oxidized pour les sauvegardes de config, weathermaps pour la visualisation

Sauvegarde critique : dump MySQL quotidien + archive RRD + configuration Nginx/PHP — rotation 30 jours minimum, tester la restauration mensuellement

Pour aller plus loin dans la sécurisation de votre infrastructure, consultez notre guide complet d'audit Active Directory 2026 qui couvre l'intégration AD avec LibreNMS et Wazuh pour une supervision complète des accès. Vous trouverez également une comparaison technique approfondie des outils d'analyse IA dans notre benchmark LLM juillet 2026, incluant les cas d'usage pour l'analyse automatique des logs de sécurité réseau.