phpIPAM : gérer les adresses IP de votre réseau avec cette solution open source

phpIPAM est la solution open source de référence pour la gestion des adresses IP (IPAM). Ce guide couvre l'installation PHP/MySQL/Apache, la découverte réseau FPING, l'import CSV, la gestion des sections, subnets et hôtes, le scan automatique et la génération de rapports.

phpIPAM est la solution open source de gestion des adresses IP (IPAM — IP Address Management) la plus déployée en entreprise, avec plus de 100 000 installations actives dans le monde selon les statistiques GitHub. La gestion des adresses IP avec phpIPAM open source remplace les fichiers Excel ou les wikis manuels par une base de données centralisée avec interface web, scan réseau automatique et API REST. phpIPAM 1.6 (version stable en 2026) supporte IPv4 et IPv6, la gestion hiérarchique des sections et sous-réseaux, la découverte automatique des hôtes via FPING/ICMP, l'import/export CSV pour les migrations, les VLANs, les VRFs (Virtual Routing and Forwarding) pour les environnements multi-tenant, les rapports personnalisables, et une API REST documentée pour l'intégration avec les outils d'automatisation (Ansible, Terraform, NetBox). L'installation s'effectue sur une stack classique PHP 8.1+/MySQL 8.0+/Apache2 ou Nginx. Ce guide pratique vous accompagne de l'installation à une configuration avancée avec scan automatique, import de plages IP existantes et mise en place des alertes d'utilisation.

À RETENIR

À retenir

Hiérarchie phpIPAM : Sections > Subnets > Hosts — les sections regroupent des sous-réseaux fonctionnels (Datacenter, Bureau, DMZ), les subnets définissent les plages IP, les hosts sont les adresses individuelles.

Scan automatique via FPING : phpIPAM utilise fping (ICMP ping de masse) pour découvrir les hôtes actifs dans chaque subnet — configurer les intervalles de scan dans les paramètres de chaque subnet, typiquement toutes les 15 minutes.

API REST intégrée : l'API phpIPAM expose des endpoints REST pour lire et modifier les données — idéal pour l'intégration avec Ansible (module phpipam) et Terraform (provider phpipam) dans les pipelines d'infrastructure-as-code.

VLANs et VRFs : phpIPAM associe les subnets à des VLANs (identifiant 802.1Q) et des VRFs pour documenter la segmentation réseau — utile pour les audits NIS 2 qui exigent une cartographie précise du réseau.

Import CSV pour migration : l'import CSV intégré supporte les colonnes IP, hostname, description, MAC — permet de migrer rapidement depuis un fichier Excel de gestion IP existant sans saisie manuelle.

Qu'est-ce que phpIPAM et pourquoi remplacer les fichiers Excel ?

La gestion des adresses IP (IPAM) est une composante critique de l'administration réseau souvent sous-estimée. Dans la majorité des PME, un fichier Excel joue le rôle d'annuaire IP — avec les problèmes inhérents : données obsolètes (personne ne met à jour quand une VM est décommissionnée), conflits d'adresses IP, aucune détection automatique des nouvelles machines,

et impossibilité d'intégration avec les outils d'automatisation. phpIPAM résout ces problèmes avec une interface centralisée, une découverte réseau automatique par scan FPING, des alertes de conflits d'IP et une API REST pour l'intégration CI/CD.

phpIPAM 1.6.0 introduit la refonte complète de l'interface (Bootstrap 5), le support PHP 8.2, la gestion améliorée des IPv6 prefixes, et l'intégration native des webhooks pour notifier des systèmes externes (Slack, Teams, Jira) lors de modifications. Par rapport à ses concurrents open source (NetBox, Nipap), phpIPAM se distingue par sa simplicité d'installation, sa faible empreinte mémoire (fonctionne sur un VPS 1 Go RAM) et son interface accessible aux équipes non-spécialistes réseau.

Installation de phpIPAM sur Debian 12 avec Apache et MySQL

L'installation de phpIPAM nécessite Apache2, PHP 8.1+ avec plusieurs extensions, MySQL 8.0+ et les packages réseau fping et nmap pour la découverte automatique.

```
 # Installation des dépendances système
apt update
apt install -y apache2 mysql-server php8.2 php8.2-mysql php8.2-curl     php8.2-gd php8.2-intl php

# Activer les modules Apache nécessaires
a2enmod rewrite
systemctl restart apache2

# Télécharger phpIPAM (version stable 1.6.x)
cd /var/www/html
git clone --depth 1 --branch 1.6.0 https://github.com/phpipam/phpipam.git phpipam
cd phpipam

# Copier et adapter la configuration
cp config.dist.php config.php
nano config.php
# Modifier :
# $db['host'] = 'localhost';
# $db['user'] = 'phpipam';
# $db['pass'] = 'VotreMotDePasse2026!';
# $db['name'] = 'phpipam';

# Créer la base de données MySQL
mysql -u root -e "
CREATE DATABASE phpipam CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
CREATE USER 'phpipam'@'localhost' IDENTIFIED BY 'VotreMotDePasse2026!';
GRANT ALL PRIVILEGES ON phpipam.* TO 'phpipam'@'localhost';
FLUSH PRIVILEGES;"

# Configurer Apache pour phpIPAM
cat > /etc/apache2/sites-available/phpipam.conf << 'EOF'

    ServerName ipam.domaine.local
    DocumentRoot /var/www/html/phpipam


        Options FollowSymLinks
        AllowOverride All
        Require all granted


    ErrorLog ${APACHE_LOG_DIR}/phpipam_error.log
    CustomLog ${APACHE_LOG_DIR}/phpipam_access.log combined
```

```
EOF

a2ensite phpipam
a2dissite 000-default
systemctl reload apache2

# Définir les permissions
chown -R www-data:www-data /var/www/html/phpipam
chmod -R 755 /var/www/html/phpipam
```

Configuration initiale et structure des sections

Après l'installation, accéder à `http://ipam.domaine.local` pour lancer l'assistant d'installation web. Choisir "Automatic database installation" pour créer automatiquement les tables. Le compte admin par défaut est admin/ipamadmin — changer immédiatement le mot de passe après la première connexion.

```
 # Structure recommandée pour une PME (à créer dans l'interface web phpIPAM)
# Sections > Subnets > Hosts

# Section 1 : Infrastructure
#   10.0.0.0/8 (supernet)
#   ├── 10.0.1.0/24  — Serveurs
#   ├── 10.0.2.0/24  — Postes de travail
#   ├── 10.0.3.0/24  — DMZ
#   ├── 10.0.10.0/24 — VoIP
#   └── 10.0.99.0/24 — Management (IPMI, iDRAC)

# Section 2 : Clients (si MSP)
#   Sous-sections par client avec VRFs séparés

# Vérification que fping est opérationnel (utilisé par le scan phpIPAM)
fping -a -g 10.0.1.0/24 2>/dev/null
# Retourne la liste des IPs répondant au ping

# Vérifier les permissions sudo pour www-data (nécessaire pour fping)
echo "www-data ALL=(ALL) NOPASSWD: /usr/bin/fping" >> /etc/sudoers.d/phpipam
chmod 440 /etc/sudoers.d/phpipam
```

Niveau	Objet phpIPAM	Exemple	Champs clés
1	Section	Infrastructure, Clients, Lab	Nom, description, permissions groupes
2	Subnet	10.0.1.0/24 — Serveurs	Réseau/masque, VLAN, VRF, gateway, scan auto
3	Host	10.0.1.10 — webserver01	IP, hostname, MAC, owner, statut, note
—	VLAN	VLAN 100 — Serveurs	ID 802.1Q, nom, description
—	VRF	VRF-Client-A	RD, nom, description

Comment configurer la découverte réseau automatique avec FPING ?

La découverte automatique des hôtes est une fonctionnalité centrale de phpIPAM. Pour chaque subnet, activer l'option "Check hosts" avec l'intervalle de scan souhaité (5 à 60 minutes).

phpIPAM utilise fping en arrière-plan pour envoyer des requêtes ICMP et détecter les nouveaux hôtes ou les hôtes qui ont cessé de répondre.

```
 # Configuration du cron pour les scans phpIPAM automatiques
# phpIPAM fournit des scripts de scan dans /var/www/html/phpipam/functions/scripts/

# Scan des subnets configurés pour la découverte automatique
*/15 * * * * www-data /usr/bin/php /var/www/html/phpipam/functions/scripts/pingCheck.php

# Résolution DNS automatique des hôtes découverts
*/30 * * * * www-data /usr/bin/php /var/www/html/phpipam/functions/scripts/resolveIPaddresses.php

# Vérification de l'état des hôtes (offline/online)
*/15 * * * * www-data /usr/bin/php /var/www/html/phpipam/functions/scripts/pingCheck.php > /dev/n

# Nettoyage des anciennes données de scan
0 3 * * * www-data /usr/bin/php /var/www/html/phpipam/functions/scripts/maintenance.php

# Vérification manuelle du scan depuis la ligne de commande
sudo -u www-data php /var/www/html/phpipam/functions/scripts/pingCheck.php
# Affiche les hôtes détectés et leur statut (online/offline)
```

Import CSV de plages IP existantes

Pour migrer depuis un fichier Excel de gestion IP vers phpIPAM, l'import CSV intégré est le moyen le plus rapide. phpIPAM accepte un format CSV flexible avec mapping de colonnes personnalisable.

```
 # Format CSV recommandé pour l'import phpIPAM
# Colonnes : ip_addr, hostname, description, mac_addr, owner, note

cat > /tmp/import_serveurs.csv << 'EOF'
ip_addr,hostname,description,mac_addr,owner,note
10.0.1.10,webserver01,Apache web server frontal,00:1A:2B:3C:4D:5E,Équipe Infra,Prod
10.0.1.11,webserver02,Apache web server frontal,00:1A:2B:3C:4D:5F,Équipe Infra,Prod
10.0.1.20,db-primary,MySQL 8.0 primaire,00:1A:2B:3C:4D:60,DBA Team,Prod - Réplication
10.0.1.21,db-replica,MySQL 8.0 réplica,00:1A:2B:3C:4D:61,DBA Team,Prod - Read-only
10.0.1.30,monitoring01,Zabbix 7.0,00:1A:2B:3C:4D:62,Monitoring,NOC
EOF

# L'import se fait via l'interface web :
# Administration > Import > Import from CSV file
# Sélectionner le subnet destination, mapper les colonnes, importer

# Via l'API phpIPAM (alternative programmatique)
# Récupérer un token d'authentification API
TOKEN=$(curl -s -X POST     -H "Content-Type: application/json"     -d '{"app_id":"mon_app","toke

echo "Token API : $TOKEN"

# Ajouter une adresse IP via l'API
curl -s -X POST     -H "token: $TOKEN"     -H "Content-Type: application/json"     -d '{"ip":"10.
```

> En déploiement phpIPAM pour un opérateur télécom régional (8 000 adresses IP actives, 15 VLANs), la migration depuis des fichiers Excel épars a pris 3 jours. Le script Python de consolidation des 12 fichiers Excel en un CSV unifié a été la partie la plus chronophage — chaque équipe avait ses propres conventions de nommage. La découverte réseau automatique a révélé 340 adresses IP actives non documentées dans les fichiers Excel, dont 12 serveurs complètement oubliés datant de 2018. Ce type de découverte justifie à elle seule l'investissement dans un outil IPAM.
>
> *— Retour de mission déploiement phpIPAM, février 2026*

Rapports et alertes phpIPAM

phpIPAM intègre un module de rapports permettant de visualiser l'utilisation des adresses IP, les subnets proches de saturation, les hôtes inactifs depuis longtemps et les conflits d'IP potentiels. Les rapports sont accessibles via Administration > Reports & tools.

```
 # Génération de rapports via l'API phpIPAM
# Liste des subnets avec leur taux d'utilisation

# Récupérer tous les subnets et leur utilisation
curl -s -H "token: $TOKEN"     http://ipam.domaine.local/api/mon_app/subnets/ |     python3 -c "
import sys, json
data = json.load(sys.stdin)['data']
for s in data:
    used = int(s.get('usage', {}).get('used', 0))
    total = int(s.get('usage', {}).get('total', 1))
    pct = round(used/total*100, 1)
    print(f"{s['subnet']}/{s['mask']} — {pct}% ({used}/{total}) — {s.get('description','')}")
"

# Alertes email pour les subnets > 80% d'utilisation
# Configurer dans Administration > phpIPAM settings > Threshold for subnet usage warnings
# Valeur recommandée : 80% (alerte préventive avant saturation)

# Script cron pour alertes personnalisées
cat > /etc/cron.d/phpipam-alerts << 'EOF'
# Vérification quotidienne des subnets proches de saturation
0 8 * * * www-data /usr/bin/php /var/www/html/phpipam/functions/scripts/alertUsersAboutUsage.php
EOF
```

Sécurisation et intégration LDAP de phpIPAM

En production, phpIPAM doit être accessible uniquement via HTTPS et les accès doivent être contrôlés via Active Directory ou LDAP. L'intégration LDAP évite de gérer des comptes locaux phpIPAM et permet d'appliquer les politiques de mot de passe de l'entreprise.

```
 # Sécuriser Apache avec HTTPS (Let's Encrypt)
apt install -y certbot python3-certbot-apache
certbot --apache -d ipam.domaine.com

# Activer la redirection HTTP vers HTTPS
# Certbot le configure automatiquement

# Restriction d'accès réseau (n'autoriser que le réseau interne)
cat >> /etc/apache2/sites-available/phpipam-le-ssl.conf << 'EOF'

        Require ip 10.0.0.0/8 192.168.0.0/16

EOF
systemctl reload apache2

# Configuration LDAP dans phpIPAM
# Administration > Authentication methods > Add LDAP authentication
# Paramètres :
# - Type : OpenLDAP
# - Host : ldap://ldap.domaine.local
# - Base DN : ou=users,dc=domaine,dc=local
# - LDAP filter : (objectClass=posixAccount)
# - Bind DN : cn=readonly,ou=services,dc=domaine,dc=local
# - Bind password : MotDePasseReadonly
```

La documentation officielle de phpIPAM est disponible sur phpipam.net/documents/installation/. Pour les aspects de cartographie réseau dans un contexte de conformité réglementaire, la méthode ANSSI de cartographie du système d'information détaille les exigences documentaires. phpIPAM s'inscrit naturellement dans une démarche de gestion des actifs conforme à ISO 27001 (contrôle A.8.1) — voir l'article sur l'inventaire des actifs en environnement cloud-native et la détection réseau NDR qui s'appuie sur une cartographie IP précise.

Questions fréquentes

Quelle est la différence entre phpIPAM et NetBox ?

NetBox (développé par DigitalOcean, maintenant open source) est plus complet que phpIPAM : il inclut la gestion des racks, câbles, circuits WAN, power panels et intégrations DCIM, en plus de l'IPAM. phpIPAM est plus léger, plus simple à déployer et à prendre en main, et suffisant pour la grande majorité des PME et ETI. NetBox est recommandé pour les datacenters et les opérateurs télécoms qui ont besoin d'une documentation d'infrastructure complète. NetBox utilise Django/Python et PostgreSQL ; phpIPAM utilise PHP et MySQL — phpIPAM a donc une stack plus courante dans les environnements PHP existants.

Comment intégrer phpIPAM avec Ansible pour l'automatisation réseau ?

La collection Ansible `phpipam.phpipam` (disponible sur Ansible Galaxy) fournit des modules pour interroger et modifier phpIPAM depuis les playbooks. Exemple d'usage : récupérer la prochaine adresse IP disponible dans un subnet avant de déployer une VM (`phpipam_address: network_address="10.0.1.0/24" state=present`). L'API REST phpIPAM est également consommable directement via le module `uri` d'Ansible pour les cas non couverts par la collection officielle. L'intégration avec Terraform est possible via le provider Terraform phpIPAM (GitHub: lord-kyron/terraform-provider-phpipam).

phpIPAM peut-il gérer les adresses IPv6 ?

Oui, phpIPAM 1.5+ supporte nativement IPv6. Les subnets IPv6 se créent comme les subnets IPv4, en notation CIDR (ex: 2001:db8::/32). La découverte automatique via fping fonctionne également en IPv6 (`fping6`). Le principal défi d'IPv6 dans phpIPAM est la gestion des espaces d'adressage très larges (/48 = 65 536 sous-réseaux /64) — il est recommandé de ne documenter que les sous-réseaux effectivement alloués plutôt que de gérer l'espace entier.

Comment sauvegarder et restaurer phpIPAM ?

La sauvegarde phpIPAM se résume à deux éléments : la base de données MySQL (`mysqldump phpipam > backup.sql`) et le fichier de configuration `config.php` . Les fichiers PHP de l'application sont récupérables depuis GitHub. Pour restaurer : installer phpIPAM, copier `config.php` , et restaurer la base de données (`mysql phpipam < backup.sql`). Il est recommandé d'automatiser cette sauvegarde avec le script mysqldump + rsync vers un serveur distant, comme décrit dans l'article sur les stratégies de sauvegarde Linux.

Peut-on utiliser phpIPAM avec un proxy inverse Nginx ?

Oui, phpIPAM fonctionne parfaitement derrière Nginx en proxy inverse. La configuration Nginx doit activer `proxy_set_header X-Forwarded-For` , `proxy_set_header X-Real-IP` , et `proxy_set_header Host` pour que phpIPAM enregistre les vraies IPs des clients. Dans `config.php` , activer `$config['proxy'] = 1;` pour que phpIPAM utilise les headers X-Forwarded-For. La gestion des sessions PHP derrière proxy nécessite également de vérifier que `session.cookie_secure = 1` si phpIPAM est servi en HTTPS.

Utilisation de l'API REST phpIPAM avec Python

L'API REST phpIPAM permet d'intégrer la gestion IP dans les workflows d'automatisation. Python est le langage le plus couramment utilisé pour consommer l'API phpIPAM, notamment dans les pipelines Ansible et les scripts de provisionning d'infrastructure. L'authentification se fait par token JWT émis lors du login, valide pendant 360 secondes par défaut (configurable).

```
 # Client Python simple pour l'API phpIPAM
cat > /opt/scripts/phpipam_client.py << 'PYEOF'
import requests

BASE_URL = 'https://ipam.domaine.local'
APP_ID   = 'mon_app'

def get_token(username, password):
    r = requests.post(f'{BASE_URL}/api/{APP_ID}/user/',
                      auth=(username, password))
    return r.json()['data']['token']

def get_next_free_ip(token, subnet_id):
    r = requests.get(f'{BASE_URL}/api/{APP_ID}/subnets/{subnet_id}/first_free/',
                     headers={'token': token})
    return r.json().get('data')

def register_ip(token, ip, subnet_id, hostname, description=''):
    data = {'ip': ip, 'subnetId': subnet_id,
            'hostname': hostname, 'description': description}
    r = requests.post(f'{BASE_URL}/api/{APP_ID}/addresses/',
                      headers={'token': token}, json=data)
    return r.json()

if __name__ == '__main__':
    token = get_token('admin', 'MotDePasse2026!')
    ip    = get_next_free_ip(token, subnet_id=5)
    print(f'Prochaine IP disponible : {ip}')
    result = register_ip(token, ip, 5, 'new-server.domaine.local')
    print(f'Enregistrement : {result}')
PYEOF

python3 /opt/scripts/phpipam_client.py
```

Intégration phpIPAM avec Terraform (infrastructure-as-code)

Le provider Terraform phpIPAM (registre Terraform) permet de réserver automatiquement des adresses IP lors du provisionning d'infrastructure. Quand Terraform crée une VM, l'adresse IP est automatiquement réservée dans phpIPAM et libérée lors de la destruction. Cette intégration élimine les conflits d'IP dans les environnements dynamiques (cloud hybride, Kubernetes).

```
 # Configuration Terraform avec provider phpIPAM
cat > main.tf << 'EOF'
terraform {
  required_providers {
    phpipam = {
      source  = "lord-kyron/phpipam"
      version = "~> 1.5"
    }
  }
}

provider "phpipam" {
  server   = "https://ipam.domaine.local"
  username = var.phpipam_user
  password = var.phpipam_password
  app_id   = "terraform"
}

data "phpipam_subnet" "servers" {
  subnet_address = "10.0.1.0"
  subnet_mask    = 24
}

resource "phpipam_first_free_address" "vm_ip" {
  subnet_id   = data.phpipam_subnet.servers.subnet_id
  hostname    = "terraform-vm-01.domaine.local"
  description = "Provisionné par Terraform"
}

output "vm_ip" {
  value = phpipam_first_free_address.vm_ip.ip_address
}
EOF

terraform init && terraform plan && terraform apply
```

phpIPAM s'inscrit dans une approche globale de gestion des actifs réseau conforme aux exigences NIS 2 (article 21 — mesures de gestion des risques de sécurité réseau). La cartographie précise des adresses IP est un prérequis pour les audits de sécurité — voir l'article sur les outils d'audit de sécurité réseau. L'intégration phpIPAM dans les pipelines CI/CD s'inscrit dans une démarche DevSecOps pour la sécurité des pipelines et la documentation d'infrastructure qui facilite les opérations NDR de détection réseau.

Mise à jour et maintenance de phpIPAM

La mise à jour de phpIPAM vers les nouvelles versions nécessite une procédure précise pour préserver les données et éviter les régressions. phpIPAM inclut des scripts de migration de base de données qui s'exécutent automatiquement lors de la première connexion après une mise à jour de version majeure. Il est indispensable de sauvegarder la base de données MySQL avant toute mise à jour.

```
 # Procédure de mise à jour phpIPAM
# 1. Sauvegarder la base de données MySQL
mysqldump phpipam | gzip > /backup/phpipam-$(date +%Y%m%d).sql.gz

# 2. Sauvegarder le fichier de configuration
cp /var/www/html/phpipam/config.php /backup/config.php.bak

# 3. Mettre à jour depuis Git
cd /var/www/html/phpipam
git fetch --tags
git checkout v1.6.1  # Remplacer par la dernière version stable

# 4. Restaurer le fichier de configuration
cp /backup/config.php.bak /var/www/html/phpipam/config.php

# 5. Vérifier les permissions
chown -R www-data:www-data /var/www/html/phpipam

# 6. Accéder à phpIPAM -> Administration -> Upgrade
# La migration de la base de données s'exécute automatiquement

# Vérification post-mise à jour
curl -s https://ipam.domaine.local/ | grep -i 'phpipam v'
php /var/www/html/phpipam/functions/scripts/pingCheck.php 2>&1 | head -5
```

phpIPAM et gestion des VLANs et VRFs

phpIPAM supporte la documentation des VLANs IEEE 802.1Q et des VRFs (Virtual Routing and Forwarding) pour les environnements multi-tenant ou avec plusieurs domaines de routage. Cette fonctionnalité est particulièrement utile pour les MSP (Managed Service Providers) qui gèrent l'espace d'adressage de plusieurs clients sur la même infrastructure physique, ou pour les entreprises avec des réseaux segmentés complexes (production, développement, DMZ, management).

```
 # Gestion des VLANs et VRFs via l'API phpIPAM
TOKEN='VOTRE_TOKEN_API'
BASE='https://ipam.domaine.local'

# Créer un VLAN
curl -s -X POST \
    -H "token: $TOKEN" \
    -H 'Content-Type: application/json' \
    $BASE/api/mon_app/vlan/ \
    -d '{"vlanId":"100","name":"Serveurs-Production",
         "description":"VLAN 100 - Serveurs de production"}'

# Créer un VRF
curl -s -X POST \
    -H "token: $TOKEN" \
    -H 'Content-Type: application/json' \
    $BASE/api/mon_app/vrf/ \
    -d '{"name":"VRF-Production","rd":"65000:100",
         "description":"VRF de production principale"}'

# Associer un subnet à un VLAN et un VRF
curl -s -X PATCH \
    -H "token: $TOKEN" \
    -H 'Content-Type: application/json' \
    $BASE/api/mon_app/subnets/5/ \
    -d '{"vlanId":"1","vrfId":"1"}'  # IDs du VLAN et VRF créés

# Lister tous les VLANs
curl -s -H "token: $TOKEN" $BASE/api/mon_app/vlan/ | python3 -m json.tool
```

La gestion rigoureuse des VLANs et VRFs dans phpIPAM simplifie les audits de conformité réseau imposés par ISO 27001 (contrôle A.13 — Sécurité des communications) et NIS 2 (article 21 — segmentation réseau). Pour des architectures réseau plus complexes incluant BGP et MPLS, des outils comme NetBox sont plus adaptés. En complément de phpIPAM, l'article sur la détection d'anomalies réseau par IA présente comment l'inventaire IP précis améliore la qualité des alertes de sécurité réseau. La cartographie IP est également un prérequis pour les tests d'intrusion réseau — voir l'article sur les outils d'audit réseau.

phpIPAM, en tant que solution IPAM open source, s'intègre dans un écosystème d'outils réseau complémentaires. Pour les environnements où la gestion des adresses IP doit être corrélée avec les flux réseau en temps réel, l'article sur la détection réseau NDR présente comment une cartographie IP précise améliore la qualité des alertes de sécurité réseau. phpIPAM constitue également un composant important pour les inventaires d'actifs exigés par les référentiels de conformité ISO 27001 et NIS 2.
