Guide Trivy - Scanner de Sécurité

Table des matières

1. Introduction à Trivy {#introduction}

Qu'est-ce que Trivy ?

Trivy est un scanner de sécurité open-source développé par Aqua Security. Il permet de détecter les vulnérabilités dans :

- Les images de conteneurs
- Les systèmes de fichiers
- Les dépôts Git
- Les images de machines virtuelles
- Les manifestes Kubernetes
- Les fichiers de configuration Infrastructure as Code (Terraform, CloudFormation, etc.)

Principales caractéristiques

- **Multi-format** : Supporte Docker, OCI, Podman, containerd
- **Multi-language** : Détecte les vulnérabilités dans les packages de nombreux langages
- **Rapide** : Analyse locale sans nécessiter de connexion réseau constante
- **Précis** : Base de données de vulnérabilités régulièrement mise à jour
- **Facile à utiliser** : Interface en ligne de commande simple

2. Installation {#installation}

Sur Linux (Ubuntu/Debian)

```
# Méthode 1 : Via le script d'installation
curl -sfL
https://raw.githubusercontent.com/aquasecurity/trivy/main/contrib/in
stall.sh | sh -s -- -b /usr/local/bin

# Méthode 2 : Via APT
sudo apt-get update
sudo apt-get install wget apt-transport-https gnupg lsb-release
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key
| sudo apt-key add -
echo "deb https://aquasecurity.github.io/trivy-repo/deb
$(lsb_release -sc) main" | sudo tee
/etc/apt/sources.list.d/trivy.list
sudo apt-get update
sudo apt-get install trivy
```

Sur macOS

```
# Via Homebrew
brew install trivy

# Via MacPorts
sudo port install trivy
```

Sur Windows

```
# Via Chocolatey
choco install trivy

# Via Scoop
scoop install trivy
```

Via Docker

```
# Utiliser Trivy directement depuis Docker
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock \
  -v $HOME/Library/Caches:/root/.cache/ aquasec/trivy:latest
```

Installation depuis les binaires

```
# Télécharger depuis GitHub Releases
wget
https://github.com/aquasecurity/trivy/releases/download/v0.45.0/trivy_0.45.0_Linux-64bit.tar.gz
tar -xzf trivy_0.45.0_Linux-64bit.tar.gz
sudo mv trivy /usr/local/bin/
```

3. Utilisation de base {#utilisation-de-base}

Première utilisation

```
# Vérifier l'installation
trivy --version

# Mettre à jour la base de données de vulnérabilités
trivy db update
```

Scanner une image Docker

```
# Scanner une image publique
trivy image nginx:latest

# Scanner une image locale
trivy image mon-app:v1.0

# Scanner avec un format de sortie spécifique
trivy image --format json nginx:latest > scan-results.json
```

Scanner un système de fichiers

```
# Scanner le répertoire courant
trivy fs .

# Scanner un répertoire spécifique
trivy fs /path/to/project

# Exclure certains dossiers
trivy fs --skip-dirs node_modules,vendor .
```

Scanner un dépôt Git

```
# Scanner un dépôt Git distant
trivy repo https://github.com/user/repo

# Scanner un dépôt local
trivy repo .
```

4. Fonctionnalités avancées {#fonctionnalités-avancées}

Filtrage par sévérité

```
# Afficher seulement les vulnérabilités critiques et élevées
trivy image --severity HIGH,CRITICAL nginx:latest

# Ignorer les vulnérabilités de faible impact
trivy image --ignore-unfixed nginx:latest
```

Formats de sortie

```
# JSON
trivy image --format json nginx:latest

# Table (par défaut)
trivy image --format table nginx:latest
```

```
# SARIF (pour intégration CI/CD)
trivy image --format sarif nginx:latest

# Template personnalisé
trivy image --format template --template "@template.tpl"
nginx:latest
```

Configuration via fichier

```
# trivy.yaml
format: json
output: results.json
severity:
  - HIGH
  - CRITICAL
ignore-unfixed: true
skip-dirs:
  - node_modules
  - vendor
 # Utiliser le fichier de configuration
trivy --config trivy.yaml image nginx:latest
```

Intégration CI/CD

```
# Fail si des vulnérabilités critiques sont trouvées
trivy image --exit-code 1 --severity CRITICAL nginx:latest

# Scanner et générer un rapport pour CI
trivy image --format sarif --output results.sarif nginx:latest
```

5. TP Pratique {#tp-pratique}

Exercice 1 : Installation et premiers pas

Objectif : Installer Trivy et effectuer votre premier scan

Étapes :

1. Installez Trivy sur votre système

2. Vérifiez l'installation avec `trivy --version`
3. Mettez à jour la base de données : `trivy db update`
4. Scannez l'image `nginx:1.20` et analysez les résultats

Questions :

- Combien de vulnérabilités sont détectées ?
- Quelle est la vulnérabilité la plus critique ?
- Quels packages sont affectés ?

Exercice 2 : Comparaison d'images

Objectif : Comparer la sécurité de différentes versions d'images

Étapes :

1. Scannez les images suivantes :
    a. `nginx:1.20`
    b. `nginx:1.21`
    c. `nginx:latest`
2. Comparez les résultats
3. Générez des rapports JSON pour chaque image

Commandes :

ICI

Questions :

- Quelle version a le moins de vulnérabilités ?
- Y a-t-il des vulnérabilités communes entre les versions ?

Exercice 3 : Scanner un projet réel

Objectif : Scanner un projet avec des dépendances

Préparation :

```
# Créer un projet Node.js de test
mkdir trivy-test-project
cd trivy-test-project
npm init -y
npm install express@4.17.1 lodash@4.17.20
```

Étapes :

1. Scannez le projet avec `trivy fs .`
2. Identifiez les vulnérabilités dans les dépendances
3. Mettez à jour les packages vulnérables
4. Re-scannez pour vérifier les améliorations

Questions :

- Quelles dépendances sont vulnérables ?
- Comment résoudre ces vulnérabilités ?

Exercice 4 : Configuration avancée

Objectif : Utiliser les fonctionnalités avancées de Trivy

Étapes :

1. Créez un fichier `.trivyignore` avec :`CVE-2021-44228`
   `CVE-2021-45046`

2. Scannez une image en ignorant ces CVE
3. Configurez Trivy pour afficher seulement les vulnérabilités HIGH et CRITICAL
4. Générez un rapport personnalisé

Commandes
ICI

Exercice 5 : Intégration dans un pipeline

Objectif : Simuler l'intégration de Trivy dans un pipeline CI/CD

Scénario : Vous devez configurer Trivy pour qu'il échoue si des vulnérabilités critiques sont trouvées

Script à créer (`security-check.sh`) :

Étapes :

1. Créez le script de sécurité
2. Rendez-le exécutable : `chmod +x security-check.sh`
3. Testez avec différentes images :`./security-check.sh nginx:latest`
   `./security-check.sh alpine:latest`

Exercice 6 : Analyse de configuration IaC

Objectif : Scanner des fichiers de configuration Infrastructure as Code

Préparation - Créez un fichier `docker-compose.yml` :
ICI

Étapes :

1. Scannez le fichier de configuration :`trivy config docker-compose.yml`

2. Identifiez les problèmes de sécurité
3. Corrigez les configurations
4. Re-scannez pour valider

Questions :

- Quels problèmes de sécurité sont détectés ?
- Comment sécuriser les secrets dans Docker Compose ?

Exercice 7 : Monitoring et alertes

Objectif : Mettre en place un système de monitoring des vulnérabilités

Script de monitoring (`monitor.sh`) :

ICI

Questions de réflexion :

- Comment automatiser ce monitoring ?
- Quels métriques surveiller ?
- Comment intégrer avec votre système d'alertes existant ?

Solutions et bonnes pratiques

Réponses aux exercices

Exercice 1 :

- Le nombre de vulnérabilités varie selon la version et la date du scan
- Consultez la section "Summary" pour un aperçu rapide

- Les packages système (comme glibc, openssl) sont souvent affectés

Exercice 2 :

- Les versions plus récentes ont généralement moins de vulnérabilités
- Utilisez `jq` pour comparer les fichiers JSON
- La version `latest` n'est pas toujours la plus sécurisée

Exercice 3 :

- Les anciennes versions de packages Node.js sont souvent vulnérables
- Utilisez `npm audit fix` pour corriger automatiquement
- Vérifiez les breaking changes avant de mettre à jour

Bonnes pratiques

1. **Automatisation** : Intégrez Trivy dans vos pipelines CI/CD
2. **Mise à jour régulière** : Mettez à jour la base de données quotidiennement
3. **Seuils de sécurité** : Définissez des seuils acceptables (ex: 0 CRITICAL)
4. **Images de base** : Utilisez des images de base sécurisées (distroless, alpine)
5. **Monitoring continu** : Scannez régulièrement vos images en production
6. **Documentation** : Documentez vos processus de sécurité
7. **Formation** : Formez vos équipes aux bonnes pratiques de sécurité

Intégrations utiles

- **GitHub Actions** : Utilisez l'action officielle Trivy
- **GitLab CI** : Intégrez dans vos jobs de pipeline
- **Jenkins** : Utilisez le plugin Trivy
- **Kubernetes** : Déployez Trivy Operator pour un scanning continu
- **Harbor** : Intégration native avec Trivy

Ce guide vous donne une base solide pour utiliser Trivy efficacement dans vos projets. N'hésitez pas à adapter les exemples à votre environnement spécifique.