Rapport : Analyse de vulnérabilités d'une image Docker (Trivy + Dockle)

1. Objectif du projet

Ce mini-projet vise à analyser la sécurité d'une image Docker simple construite localement. L'objectif est double :

- **Identifier les vulnérabilités connues** (CVE) dans l'image avec **Trivy**,
- **Évaluer les bonnes pratiques de construction** d'image avec **Dockle**.

2. Environnement

- **Système** : Ubuntu 22.04 (WSL ou VM)
- **Outils utilisés** :
  - `Docker`
  - `Trivy`
  - `Dockle`
- **Image testée** : `sec-test:1.0`, construite localement

3. Dockerfile utilisé

```
FROM python:3.9-slim

USER root


RUN set -eux; \
    apt-get update; \
    apt-get install -y --no-install-recommends \
        ca-certificates \
        netbase \
        tzdata \
        curl; \
    apt-get clean; \
    rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*


RUN useradd -m appuser
USER appuser

WORKDIR /home/appuser
COPY app.py .

HEALTHCHECK CMD curl --fail http://localhost:8000 || exit 1
CMD ["python", "app.py"]
```

4. Scan avec Trivy

Commande :
trivy image sec-test:1.0

Résultats :

- **Système** : Debian 13.1
- **Nombre total de vulnérabilités** : 61
  - **Critiques** : 0
  - **Hautes** : 0
  - **Moyennes** : 10
  - **Faibles** : 51
- Quelques paquets vulnérables : `glibc`, `util-linux`, `apt`, `pip`

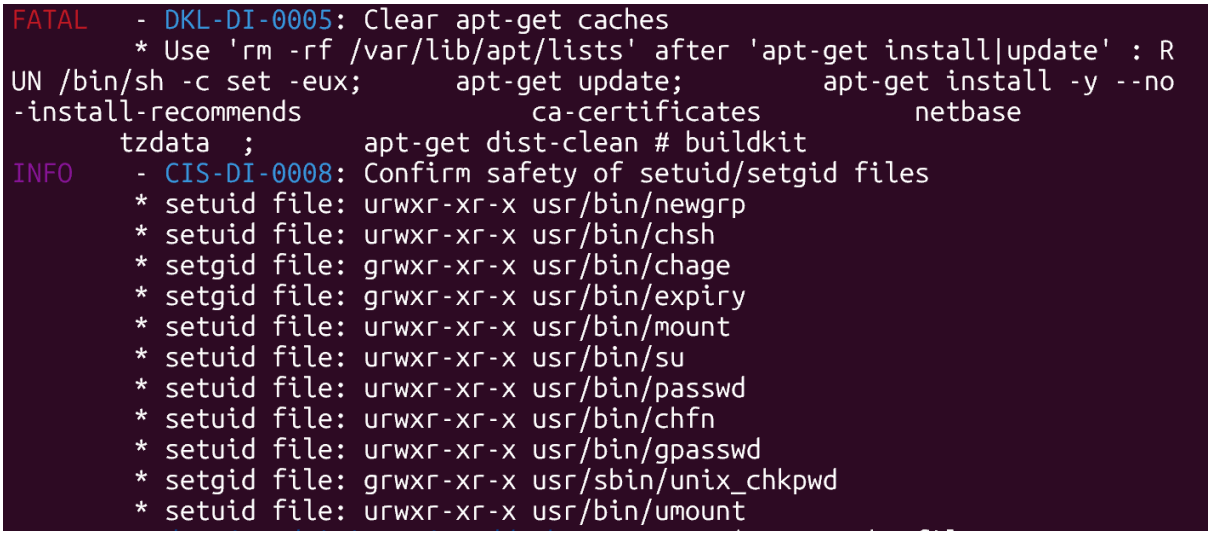
Interprétation :

L'image contient des **vulnérabilités faibles à moyennes**, fréquentes dans les images de base, mais **aucune critique**. La présence de `pip` vulnérable (CVE-2023-5752) peut poser problème si le conteneur est exposé.

5. Audit avec Dockle

Commande :
dockle sec-test:1.0

Résultat principal :



- **FATAL - DKL-DI-0005** : Cache `apt` non nettoyé (malgré présence du `rm -rf`) ➜ **faux positif probable**
- **INFO - CIS-DI-0008** : Fichiers `setuid`/`setgid` présents ➜ comportement attendu pour des outils système

Bonnes pratiques détectées :

- Utilisation d'un utilisateur non-root (`appuser`)
- Healthcheck présent
- Image légère basée sur `python:3.9-slim`

6. Améliorations possibles

- Mettre à jour les dépendances vulnérables (`pip`, `glibc`, etc.)
- Supprimer plus de fichiers temporaires (si nécessaire)
- Supprimer ou désactiver les utilitaires avec `setuid` si non nécessaires
- Ajouter un fichier `requirements.txt` pour un audit plus fin des dépendances Python

7. Conclusion

Ce projet a permis de :

- Construire une image Docker personnalisée et fonctionnelle,
- Apprendre à **intégrer la sécurité dans le cycle de vie Docker (DevSecOps)**,
- Identifier les **vulnérabilités connues (CVE)** et les **bonnes/mauvaises pratiques de sécurité**.

Ce type d'analyse est essentiel avant tout déploiement, et s'intègre dans un pipeline CI/CD sécurisé.