Rendu TP01

Compte-rendu du TP01 effectué par `Thomas PEUGNET`.

Préparation

Nous mettons en place notre environnement avec le script suivant.

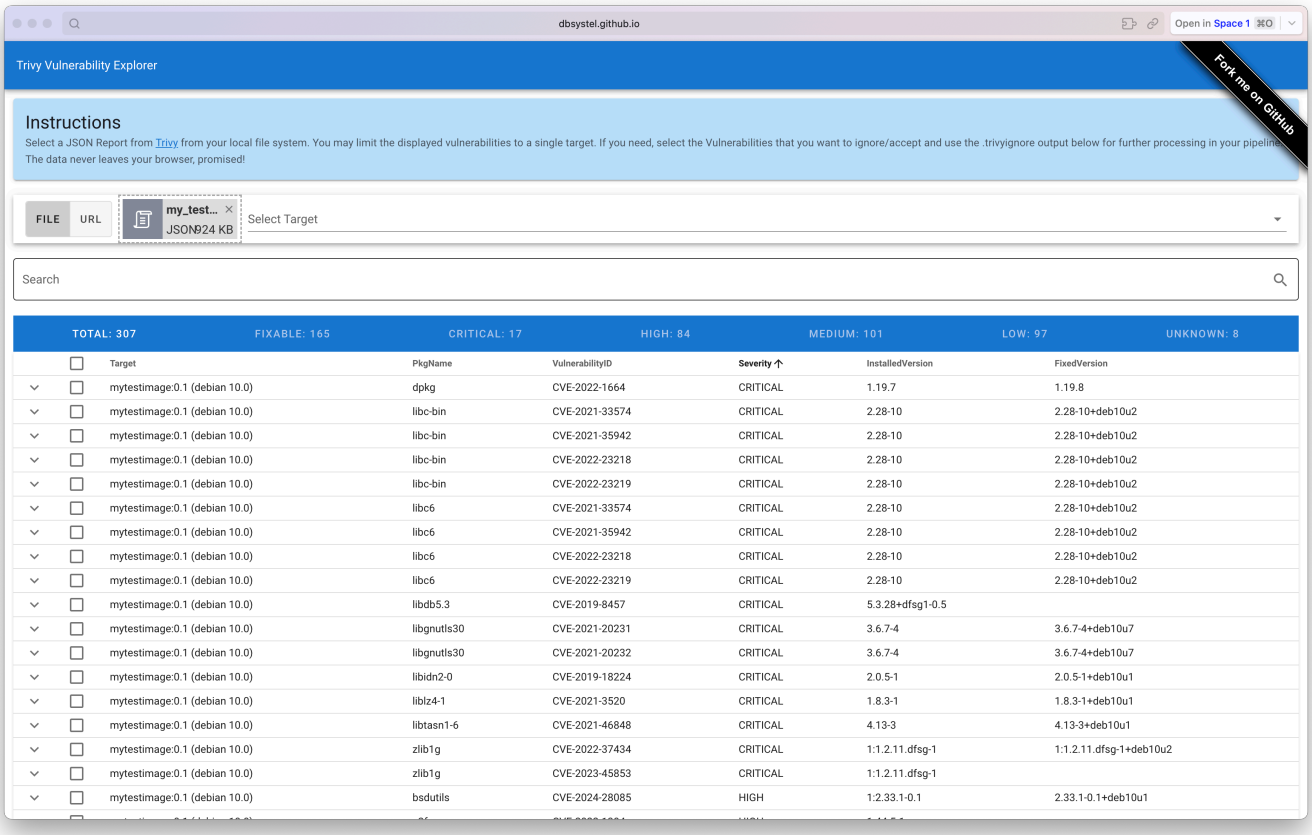
```
#!/bin/bash

# Met à jour les dépôts et met à niveau les paquets
apt update && apt upgrade -y

# Installe les outils nécessaires
apt install -y open-vm-tools net-tools ca-certificates curl

truncate -s 0 /etc/machine-id
rm /var/lib/dbus/machine-id
ln -s /etc/machine-id /var/lib/dbus/machine-id

# Installe Docker
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
chmod a+r /etc/apt/keyrings/docker.asc
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] \
https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo \"$VERSION_CODENAME\") stable" | tee
/etc/apt/sources.list.d/docker.list > /dev/null
apt-get update -y
apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-
compose-plugin

# Vérifie l'installation de Docker
docker -v

# Installe Trivy
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor |
tee /usr/share/keyrings/trivy.gpg > /dev/null
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg]
https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | tee
/etc/apt/sources.list.d/trivy.list > /dev/null
apt-get update -y
apt-get install -y trivy
trivy -h

# Crée un fichier et un Dockerfile d'exemple
mkdir archive
echo "this is some text" > ./archive/file.txt
```

```
tar cvf myfile.tar archive
cat << 'EOF' > Dockerfile
FROM debian:10.0
RUN apt-get -y install bash
ADD ./myfile.tar /tmp
EXPOSE 22
EOF

# Construit une image Docker
sudo docker build -t mytestimage:0.1 ./ -f Dockerfile

# Lance le service Docker
sudo service docker start

# Vérifie les conteneurs en cours d'exécution
docker ps

# Exécute un scan Trivy sur l'image créée
trivy image -f json -o mytestimage_results_"$(date +"%H%M%S")".txt mytestimage:0.1

# Archive un historique des fichiers générés
mkdir -p /root/scan_results
mv mytestimage_results_*.txt /root/scan_results/
echo "Scan terminé. Résultats sauvegardés dans /root/scan_results."
```

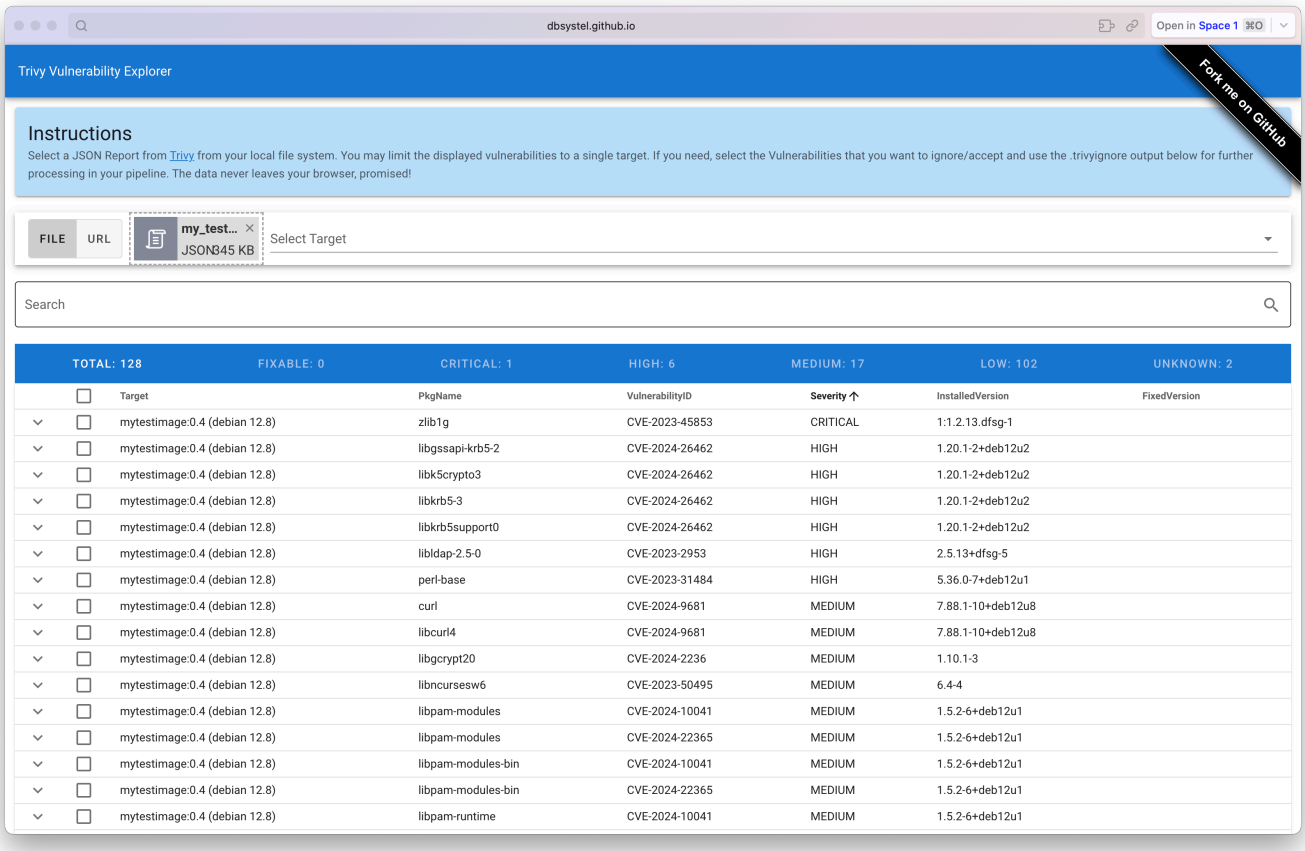
Exécuter le script en `sudo`.

Envoi des données

Nous envoyons le résultat de l'analyse sur `dbsystel.github.io`.



Nous changeons la version de Debian et faisons quelques changements sur le Dockerfile.



Notre Dockerfile, à cet instant, est le suivant.

```
FROM debian:12

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && \
    apt-get upgrade -y && \
    apt-get install -y --no-install-recommends \
    bash \
    openssh-server \
    curl \
    ca-certificates && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

ADD ./myfile.tar /tmp

RUN useradd -ms /bin/bash appuser && \
    mkdir -p /home/appuser/.ssh && \
    chmod 700 /home/appuser/.ssh

USER appuser

EXPOSE 22

CMD ["bash"]
```

Les changements que nous avons fait:

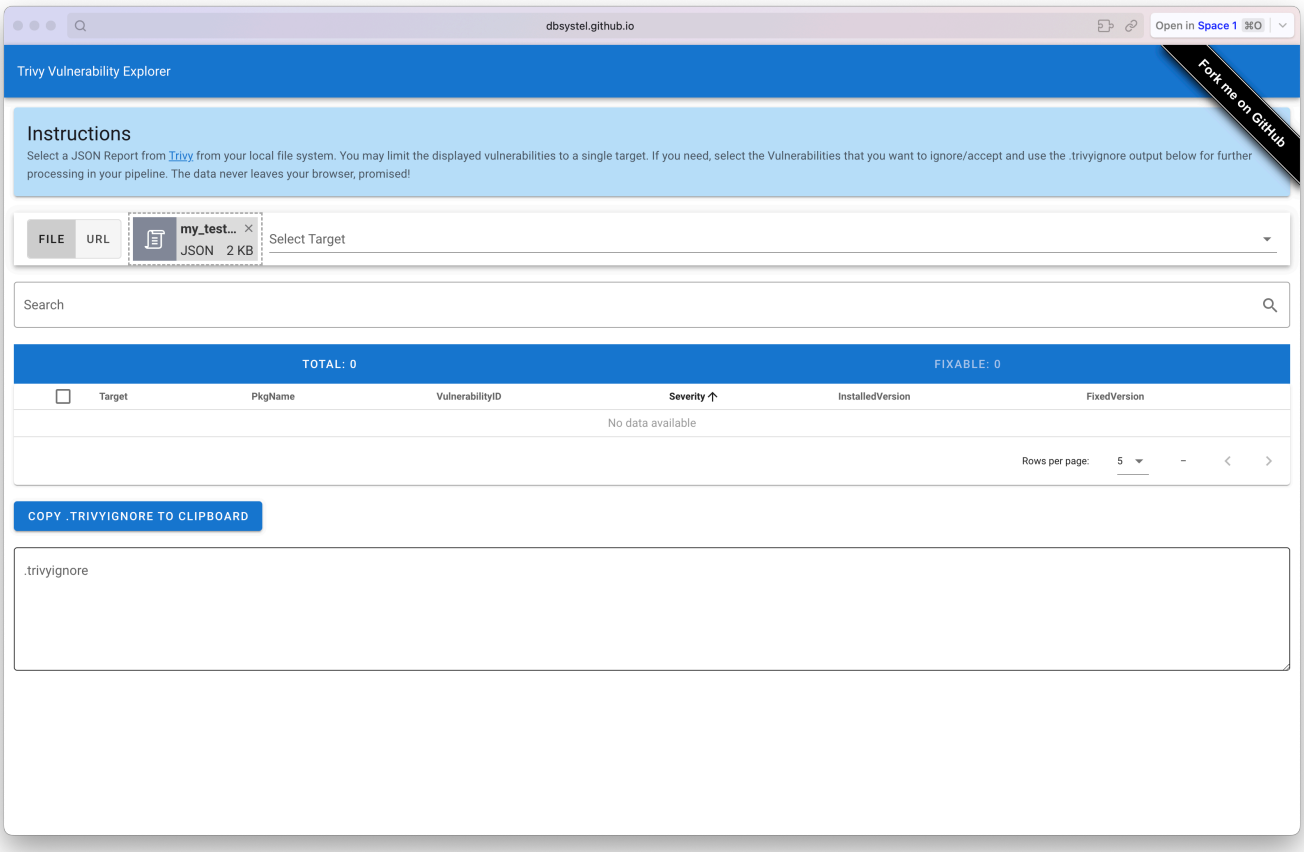
- `apt-get update` et `apt-get upgrade` pour mettre à jour les paquets, suivi d'une installation des paquets nécessaires (`bash`, `openssh-server`, `curl`, `ca-certificates`), avec nettoyage du cache pour réduire la taille de l'image.
- Plus une bonne pratique qu'une vulnérabilité: `myfile.tar` déplacé à `/tmp`.
- SSH: Création de l'utilisateur `appuser`, configuration du répertoire `.ssh` et passage à cet utilisateur pour exécuter les commandes.

Puis, nous avons décidé de passer sur une image `alpine`, moins complète donc moins vulnérable.

Notre Dockerfile est à présent le suivant:

```
FROM alpine:3.21

RUN apk add --no-cache bash

ADD ./myfile.tar /tmp

EXPOSE 22
```

Et là nous obtenons un total de 0 vulnérabilité.



Nous avons, cette fois-ci, complètement changé notre fusil d'épaule en changeant le nom de l'image. Sans changer d'OS, nous sommes toujours sur un debian et n'avons pas impacté le fonctionnement de l'image.

Nous ne l'avons pas détaillé plus haut, mais nous effectuons systématiquement les commandes suivantes:

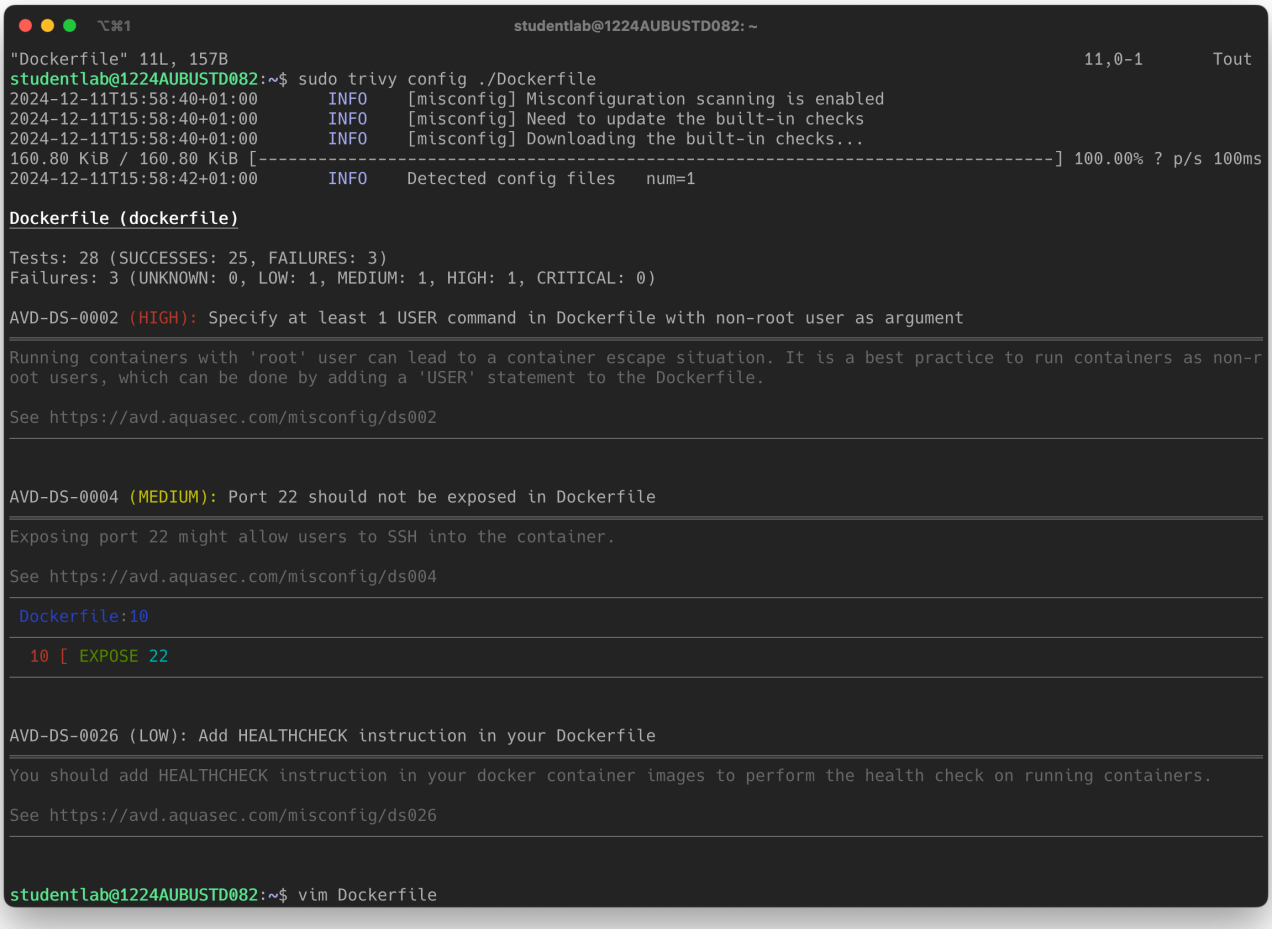
```
sudo  docker build -t mytestimage:0.6 ./ -f Dockerfile
```

```
sudo trivy image -f json -o mytestimage_result.json mytestimage:0.6
```

Nous envoyons ensuite le résultat json sur le site indiqué pour obtenir les informations sur les vulnérabilités.

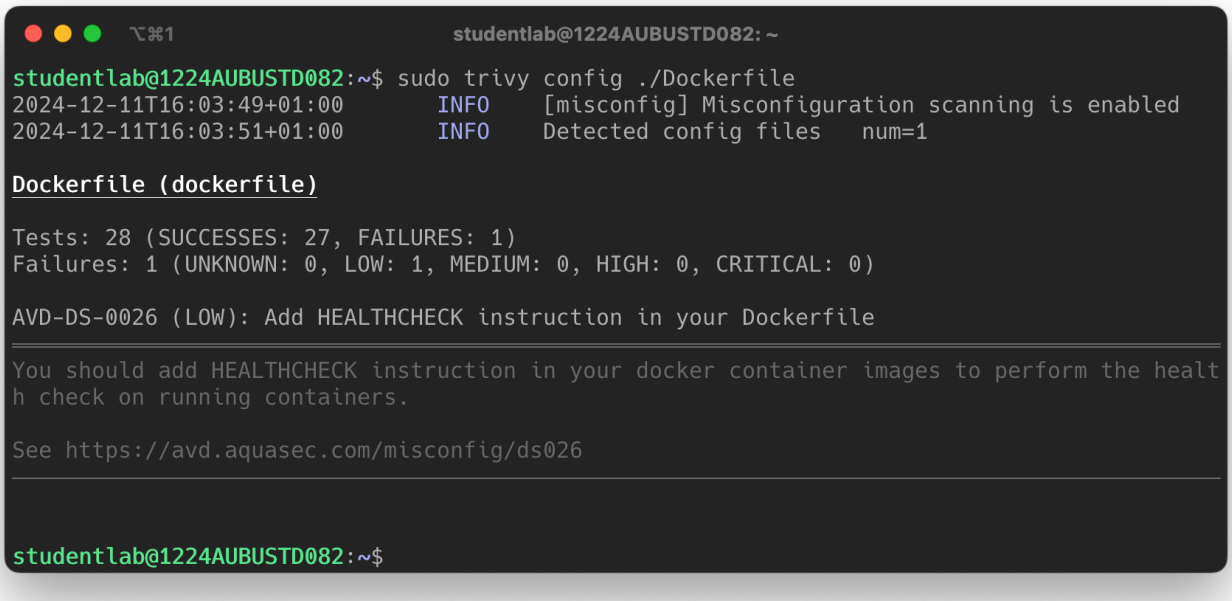
Analyse

Nous lançons la commande `trivy config Dockerfile` depuis notre répertoire `/hom/studenlab`.



Nous modifions notre Dockerfile pour avoir le contenu suivant.

```
FROM alpine:3.21

RUN apk add --no-cache bash

ADD ./myfile.tar /tmp

RUN adduser -D appuser

USER appuser

CMD ["bash"]
```



Nous obtenons une seule vulnérabilité, qui est le fait d'avoir un Healthcheck sur notre Dockerfile. Étant donné que nous ne runnons aucun service et que ce conteneur est simplement une image linux qui tourne toute seule, un Healthcheck n'est pas forcément nécessaire. De plus, le résultat demeure satisfaisant même si nous n'avons pas obtenu 0 Failures..