

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 `dbsystemel.github.io` .

Trivy Vulnerability Explorer

Instructions

Select a JSON Report from [Trivy](#) from your local file system. You may limit the displayed vulnerabilities to a single target. If you need, select the Vulnerabilities that you want to ignore/accept and use the .trivyignore output below for further processing in your pipeline. The data never leaves your browser, promised!

FILEURL

my_test...
JSON924 KB

Select Target

Search

TOTAL: 307

FIXABLE: 165

CRITICAL: 17

HIGH: 84

MEDIUM: 101

LOW: 97

UNKNOWN: 8

<input type="checkbox"/>	Target	PkgName	VulnerabilityID	Severity ↑	InstalledVersion	FixedVersion
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	dpkg	CVE-2022-1664	CRITICAL	1.19.7	1.19.8
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	libc-bin	CVE-2021-33574	CRITICAL	2.28-10	2.28-10+deb10u2
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	libc-bin	CVE-2021-35942	CRITICAL	2.28-10	2.28-10+deb10u2
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	libc-bin	CVE-2022-23218	CRITICAL	2.28-10	2.28-10+deb10u2
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	libc-bin	CVE-2022-23219	CRITICAL	2.28-10	2.28-10+deb10u2
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	libc6	CVE-2021-33574	CRITICAL	2.28-10	2.28-10+deb10u2
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	libc6	CVE-2021-35942	CRITICAL	2.28-10	2.28-10+deb10u2
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	libc6	CVE-2022-23218	CRITICAL	2.28-10	2.28-10+deb10u2
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	libc6	CVE-2022-23219	CRITICAL	2.28-10	2.28-10+deb10u2
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	libdb5.3	CVE-2019-8457	CRITICAL	5.3.28+dfsg1-0.5	
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	libgnutls30	CVE-2021-20231	CRITICAL	3.6.7-4	3.6.7-4+deb10u7
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	libgnutls30	CVE-2021-20232	CRITICAL	3.6.7-4	3.6.7-4+deb10u7
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	libidn2-0	CVE-2019-18224	CRITICAL	2.0.5-1	2.0.5-1+deb10u1
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	liblz4-1	CVE-2021-3520	CRITICAL	1.8.3-1	1.8.3-1+deb10u1
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	libtasn1-6	CVE-2021-46848	CRITICAL	4.13-3	4.13-3+deb10u1
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	zlib1g	CVE-2022-37434	CRITICAL	1:1.2.11.dfsg-1	1:1.2.11.dfsg-1+deb10u2
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	zlib1g	CVE-2023-45853	CRITICAL	1:1.2.11.dfsg-1	
<input type="checkbox"/>	mytestimage:0.1 (debian 10.0)	bsdutils	CVE-2024-28085	HIGH	1:2.33.1-0.1	2.33.1-0.1+deb10u1

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

Trivy Vulnerability Explorer

Instructions

Select a JSON Report from [Trivy](#) from your local file system. You may limit the displayed vulnerabilities to a single target. If you need, select the Vulnerabilities that you want to ignore/accept and use the .trivyignore output below for further processing in your pipeline. The data never leaves your browser, promised!

FILEURL

my_test...
JSON845 KB

Select Target

Search

TOTAL: 128

FIXABLE: 0

CRITICAL: 1

HIGH: 6

MEDIUM: 17

LOW: 102

UNKNOWN: 2

<input type="checkbox"/>	Target	PkgName	VulnerabilityID	Severity ↑	InstalledVersion	FixedVersion
<input type="checkbox"/>	mytestimage:0.4 (debian 12.8)	zlib1g	CVE-2023-45853	CRITICAL	1:1.2.13.dfsg-1	
<input type="checkbox"/>	mytestimage:0.4 (debian 12.8)	libgssapi-krb5-2	CVE-2024-26462	HIGH	1.20.1-2+deb12u2	
<input type="checkbox"/>	mytestimage:0.4 (debian 12.8)	libk5crypto3	CVE-2024-26462	HIGH	1.20.1-2+deb12u2	
<input type="checkbox"/>	mytestimage:0.4 (debian 12.8)	libkrb5-3	CVE-2024-26462	HIGH	1.20.1-2+deb12u2	
<input type="checkbox"/>	mytestimage:0.4 (debian 12.8)	libkrb5support0	CVE-2024-26462	HIGH	1.20.1-2+deb12u2	
<input type="checkbox"/>	mytestimage:0.4 (debian 12.8)	libldap-2.5-0	CVE-2023-2953	HIGH	2.5.13+dfsg-5	
<input type="checkbox"/>	mytestimage:0.4 (debian 12.8)	perl-base	CVE-2023-31484	HIGH	5.36.0-7+deb12u1	
<input type="checkbox"/>	mytestimage:0.4 (debian 12.8)	curl	CVE-2024-9681	MEDIUM	7.88.1-10+deb12u8	
<input type="checkbox"/>	mytestimage:0.4 (debian 12.8)	libcurl4	CVE-2024-9681	MEDIUM	7.88.1-10+deb12u8	
<input type="checkbox"/>	mytestimage:0.4 (debian 12.8)	libcrypt20	CVE-2024-2236	MEDIUM	1.10.1-3	
<input type="checkbox"/>	mytestimage:0.4 (debian 12.8)	libcursesw6	CVE-2023-50495	MEDIUM	6.4-4	
<input type="checkbox"/>	mytestimage:0.4 (debian 12.8)	libpam-modules	CVE-2024-10041	MEDIUM	1.5.2-6+deb12u1	
<input type="checkbox"/>	mytestimage:0.4 (debian 12.8)	libpam-modules	CVE-2024-22365	MEDIUM	1.5.2-6+deb12u1	
<input type="checkbox"/>	mytestimage:0.4 (debian 12.8)	libpam-modules-bin	CVE-2024-10041	MEDIUM	1.5.2-6+deb12u1	
<input type="checkbox"/>	mytestimage:0.4 (debian 12.8)	libpam-modules-bin	CVE-2024-22365	MEDIUM	1.5.2-6+deb12u1	
<input type="checkbox"/>	mytestimage:0.4 (debian 12.8)	libpam-runtime	CVE-2024-10041	MEDIUM	1.5.2-6+deb12u1	

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:

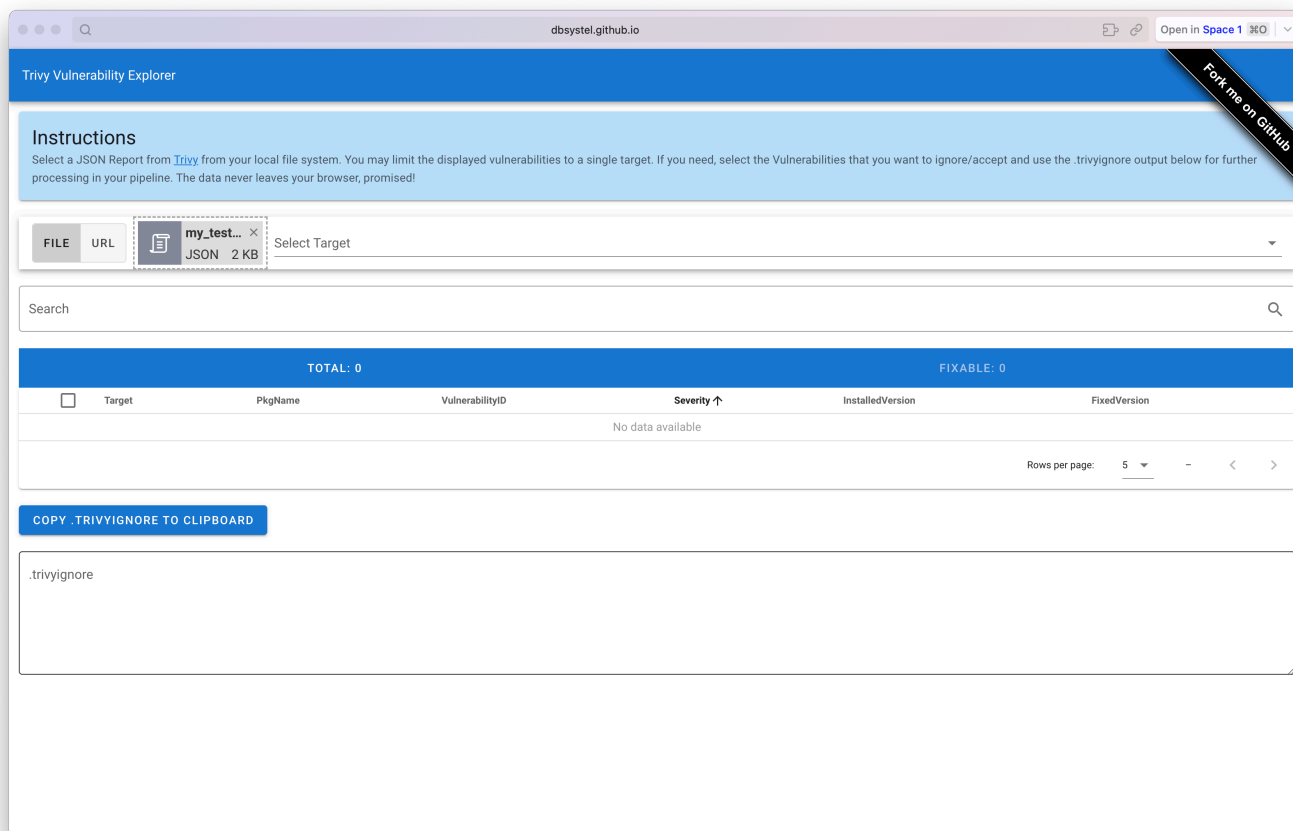
```
FROM alpine:3.21

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`.

```
studentlab@1224AUBUSTD082: ~  
"Dockerfile" 11L, 157B  
studentlab@1224AUBUSTD082:~$ sudo trivy config ./Dockerfile  
2024-12-11T15:58:40+01:00 INFO [misconfig] Misconfiguration scanning is enabled  
2024-12-11T15:58:40+01:00 INFO [misconfig] Need to update the built-in checks  
2024-12-11T15:58:40+01:00 INFO [misconfig] Downloading the built-in checks...  
160.80 KiB / 160.80 KiB [-----] 100.00% ? p/s 100ms  
2024-12-11T15:58:42+01:00 INFO Detected config files num=1  
  
Dockerfile (dockerfile)  
  
Tests: 28 (SUCCESSSES: 25, FAILURES: 3)  
Failures: 3 (UNKNOWN: 0, LOW: 1, MEDIUM: 1, HIGH: 1, CRITICAL: 0)  
  
AVD-DS-0002 (HIGH): Specify at least 1 USER command in Dockerfile with non-root user as argument  
  
Running containers with 'root' user can lead to a container escape situation. It is a best practice to run containers as non-root users, which can be done by adding a 'USER' statement to the Dockerfile.  
  
See https://avd.aquasec.com/misconfig/ds002  
  
AVD-DS-0004 (MEDIUM): Port 22 should not be exposed in Dockerfile  
  
Exposing port 22 might allow users to SSH into the container.  
  
See https://avd.aquasec.com/misconfig/ds004  
  
Dockerfile:10  
10 [ EXPOSE 22  
  
AVD-DS-0026 (LOW): Add HEALTHCHECK instruction in your Dockerfile  
  
You should add HEALTHCHECK instruction in your docker container images to perform the health check on running containers.  
  
See https://avd.aquasec.com/misconfig/ds026  
  
studentlab@1224AUBUSTD082:~$ vim Dockerfile
```

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"]
```

```
studentlab@1224AUBUSTD082: ~  
studentlab@1224AUBUSTD082:~$ sudo trivy config ./Dockerfile  
2024-12-11T16:03:49+01:00      INFO    [misconfig] Misconfiguration scanning is enabled  
2024-12-11T16:03:51+01:00      INFO    Detected config files    num=1  
  
Dockerfile (dockerfile)  
  
Tests: 28 (SUCCESES: 27, FAILURES: 1)  
Failures: 1 (UNKNOWN: 0, LOW: 1, MEDIUM: 0, HIGH: 0, CRITICAL: 0)  
  
AVD-DS-0026 (LOW): Add HEALTHCHECK instruction in your Dockerfile  
  
You should add HEALTHCHECK instruction in your docker container images to perform the health check on running containers.  
  
See https://avd.aquasec.com/misconfig/ds026  
  
studentlab@1224AUBUSTD082:~$
```

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..