Sécurité des infrastructures de virtualisation

Conteneurs et sécurité - Solutions

3e année cycle ingénieur STI, option 2SU, 2018 – 2019

- **Question 1 :** Sous quel utilisateur tourne ce shell ?
- **Réponse :** La commande id lancée dans le conteneur nous indique que ce shell tourne en root :

```
[fedora@fedora-dev ~]# docker run -it --rm fedora bash
[root@644ee121dac7 /]# id
uid=0(root) gid=0(root) groups=0(root)
```

- **Question 2 :** Est-ce le même à l'intérieur et à l'extérieur du conteneur ?
- **Réponse :** En cherchant l'identifiant du conteneur dans la sortie de la commande `ps auxf`, on peut trouver les informations sur le processus `bash` s'exécutant dans le conteneur :

```
# ps auxf | less
root 9693 0.0 0.4 287592 9244 ? Ssl 13:31 0:00 /usr/libexec/docker/docker-containerd-current --listen ↙
    ↳ unix:///run/containerd.sock --shim /usr/libexec/docker/docker-containerd-shim-current -- ↙
    ↳ start-timeout 2m
root 10036 0.0 0.1 190208 2952 ? Sl 13:34 0:00 \_ /usr/libexec/docker/docker-containerd-shim-current ↙
    ↳  644ee121dac78d572f7bbd2064cf212d0a01949946b6de3dbf77e7cd5898d61b /var/run/docker/ ↙
    ↳ libcontainerd/644ee121dac78d572f7bbd2064cf212d0a01949946b6de3dbf77e7cd5898d61b /usr/ ↙
    ↳ libexec/docker/docker-runc-current
root 10050 0.0 0.1 12588 3800 pts/1 Ss+ 13:34 0:00 \_ bash
```

L'utilisateur sous lequel tourne le shell est aussi root lorsqu'on l'observe depuis l'extérieur du conteneur.

- **Question 3 :** Sous quel contexte SELinux et avec quelles catégories tourne ce processus ?
- **Réponse :** A l'intérieur du conteneur, les processus ont l'impression que le système fonctionne sans SELinux. Vu de l'extérieur, on peut trouver un contexte de la forme suivante :

```
# ps axfZ | less
system_u:system_r:container_runtime_t:s0 9693 ? Ssl 0:00 /usr/libexec/docker/docker-containerd- ↙
    ↳ current --listen unix:///run/containerd.sock --shim /usr/libexec/docker/docker-containerd- ↙
    ↳ shim-current --start-timeout 2m
system_u:system_r:container_runtime_t:s0 10036 ? Sl 0:00 \_ /usr/libexec/docker/docker- ↙
    ↳ containerd-shim-current 644 ↙
    ↳ ee121dac78d572f7bbd2064cf212d0a01949946b6de3dbf77e7cd5898d61b /var/run/docker/ ↙
    ↳ libcontainerd/644ee121dac78d572f7bbd2064cf212d0a01949946b6de3dbf77e7cd5898d61b /usr/ ↙
    ↳ libexec/docker/docker-runc-current
system_u:system_r:container_t:s0:c212,c687 10050 pts/1 Ss+ 0:00 \_ bash
```

Les processus à l'intérieur du conteneur tournent donc sous le contexte : `system_u:system_r:container_t:s0:c212,c687`.

- **Question 4 :** Sous quel utilisateur, contexte SELinux et avec quelles catégories tourne ce processus ?
- **Réponse :** Il tourne sous le même utilisateur et le même contexte SELinux mais il ne partage pas les même catégories SELinux. Les processus de conteneurs différents ne peuvent donc pas interagir entre eux.

- **Question 5 :** Sous quel utilisateur, contexte SELinux et avec quelles catégories tourne ce processus ?
- **Réponse :** De façon similaire aux questions précédentes, on peut déterminer que les processus du conteneur tournent sous le contexte : `system_u:system_r:container_runtime_t:s0`.
- **Question 6 :** Quels sont les risques présentés par cette commande ?
- **Réponse :** Cette commande met à disposition des programmes à l'intérieur du conteneur l'ensemble du système de fichier de l'hôte. Les processus à l'intérieur du conteneur peuvent alors interagir avec le système hôte, ce qui pose des problèmes de sécurité.
- **Question Dockerfile :** Créer un conteneur Docker qui :
  — est basé sur l'image de confiance fournie par le projet Fedora ;
  — ne tourne pas sous l'utilisateur `root` ;
  — lance un serveur Apache sur le port 8080.
- **Réponse :** Suggestion de Dockerfile :

```
FROM fedora
MAINTAINER Timothée Ravier

# Mise à jour de l'ensemble de l'image et installation de Apache
RUN dnf -y update ; dnf -y install httpd ; dnf clean all

# Modification de la configuration de Apache pour qu'il ne soit plus nécessaire
# de l'exécuter en root puisque le port 8080 est non privilégié
RUN sed -i 's|Listen 80|Listen 8080|g' /etc/httpd/conf/httpd.conf

# Export du port 8080 à l'extérieur du conteneur
EXPOSE 8080

# Il n'est plus nécessaire de changer d'utilisateur
RUN sed -i 's|User apache|#User apache|g' /etc/httpd/conf/httpd.conf
RUN sed -i 's|Group apache|#Group apache|g' /etc/httpd/conf/httpd.conf

# L'installation du paquet httpd a créé l'utilisateur et le groupe apache que
# nous pouvons utiliser comme utilisateur non privilégié. Sinon, nous aurions
# pu créer un utilisateur à l'aide de la commande :
# RUN useradd -d /var/lib/http -M -r -s /sbin/nologin http

# Il n'est plus nécessaire d'être root pour ouvrir les fichiers de log
RUN chown -R apache:apache /var/log/httpd

# Le dossier /run/httpd sera monté en tmpfs
RUN rm -rf /run/httpd

# Les commandes qui suivent seront désormais lancées en tant qu'utilisateur
# apache
USER apache
CMD ["/usr/sbin/httpd", "-e", "info", "-DFOREGROUND"]
```

Construction d'une image à partir de ce Dockerfile :

```
$ ls
Dockerfile

$ sudo docker build --tag apache .
```

```
Sending build context to Docker daemon 4.096 kB
Step 1/11 : FROM docker.io/fedora
 ---> 8bd04269a51b
Step 2/11 : MAINTAINER Timothée Ravier
 ---> Running in 6faf21c79831
 ---> 1a82fd982a48
Removing intermediate container 6faf21c79831
Step 3/11 : RUN dnf -y update ; dnf -y install httpd ; dnf clean all
 ---> Running in b68f57edefeb
...
 ---> 350d7131f307
Removing intermediate container b68f57edefeb
Step 4/11 : RUN sed -i 's|Listen 80|Listen 8080|g' /etc/httpd/conf/httpd.conf
 ---> Running in b05764c6e4ee
 ---> a2ffba736b6e
Removing intermediate container b05764c6e4ee
Step 5/11 : EXPOSE 8080
 ---> Running in 4f993bea4f47
 ---> ecc1560f713b
Removing intermediate container 4f993bea4f47
Step 6/11 : RUN sed -i 's|User apache|#User apache|g' /etc/httpd/conf/httpd.conf
 ---> Running in 0f8f609f50d9
 ---> 0318c15d87cd
Removing intermediate container 0f8f609f50d9
Step 7/11 : RUN sed -i 's|Group apache|#Group apache|g' /etc/httpd/conf/httpd.conf
 ---> Running in 388a72c78525
 ---> 960360c45587
Removing intermediate container 388a72c78525
Step 8/11 : RUN chown -R apache:apache /var/log/httpd
 ---> Running in 5f4bbe83c346
 ---> 057cad7e8000
Removing intermediate container 5f4bbe83c346
Step 9/11 : RUN rm -rf /run/httpd
 ---> Running in 6c4201f6c4ac
 ---> 84b857f33a96
Removing intermediate container 6c4201f6c4ac
Step 10/11 : USER apache
 ---> Running in 7b4172e9fbfe
 ---> e948fdcf89eb
Removing intermediate container 7b4172e9fbfe
Step 11/11 : CMD /usr/sbin/httpd -e info -DFOREGROUND
 ---> Running in 137971c05733
 ---> 5bf02a41a820
Removing intermediate container 137971c05733
Successfully built 5bf02a41a820

$ mkdir ~/www ~/logs
$ echo "INSA" > ~/www/test.html
$ sudo chown -R apache: ~/www ~/logs

$ sudo docker run --detach=true --publish 8080:8080 --read-only \
      -v /home/vagrant/logs:/var/log/httpd:Z \
      -v /home/vagrant/www:/var/www/html:Z \
      --tmpfs /run/httpd:mode=777 \
      apache
64a5160e29e0fc2db13942b069975c32397b7c7f010b91543b37de61e0ccf330
```

```
$ sudo docker inspect \
      -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' \
      64a5160e29e0f...
172.17.0.2

$ curl 172.17.0.2:8080/test.html
INSA
```