Règles IPTables pour Docker

Par défaut, Docker autorise toutes les adresses IP externes à se connecter aux conteneurs, pour restreindre l'accès à une seule IP ou à un réseau, il faut ajouter une règle dans la chaîne `DOCKER-USER`, située avant celles gérées par Docker dans la chaîne `DOCKER`, sans modifier directement ces dernières. Notez que les exemples concernent l'IPv4 : pour une infrastructure en IPv6, il convient d'appliquer des règles similaires avec `ip6tables`, afin de couvrir à la fois IPv4 (`iptables`) et IPv6 (`ip6tables`).

Étapes pour appliquer les règles après le démarrage de Docker

Créez un fichier script, par exemple `/usr/local/bin/iptables-rules.sh` :

```
sudo nano /usr/local/bin/iptables-rules.sh
```

Ajouter les règles dans le script :

```
#!/bin/bash
# Remise à zéro des règles personnalisées Docker
iptables -F DOCKER-USER
ip6tables -F DOCKER-USER

# Autorise l'accès au port 9443 uniquement depuis 111.11.111.111
iptables -A DOCKER-USER -p tcp --dport 9443 -s 111.11.111.111 -j ACCEPT

# Autorise l'accès public aux ports 80 (HTTP) et 443 (HTTPS)
iptables -A DOCKER-USER -p tcp --dport 80 -j ACCEPT
iptables -A DOCKER-USER -p tcp --dport 443 -j ACCEPT

# Autorise tout le trafic sortant des containers (nécessaire pour apt/pip/npm, etc.)
iptables -A DOCKER-USER -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
iptables -A DOCKER-USER -o ens16 -j ACCEPT

# Bloque tout le reste (en dernier)
```

```
iptables -A DOCKER-USER -j DROP

# ================== Pour IPv6 ==================

# Autorise l'accès public aux ports 80 et 443 en IPv6
ip6tables -A DOCKER-USER -p tcp --dport 80 -j ACCEPT
ip6tables -A DOCKER-USER -p tcp --dport 443 -j ACCEPT

# Autorise le trafic sortant IPv6
ip6tables -A DOCKER-USER -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
ip6tables -A DOCKER-USER -o ens16 -j ACCEPT

# Bloque tout le reste
ip6tables -A DOCKER-USER -j DROP
```

Rendre le script exécutable :

```
sudo chmod +x /usr/local/bin/iptables-rules.sh
```

Ci-dessous une version qui lit un fichier de whitelist peut être parfaite pour du proxy Cloudflare :

```
#!/bin/bash

# Remise à zéro des règles personnalisées Docker
iptables -F DOCKER-USER
ip6tables -F DOCKER-USER

# Remplacer 'ext_if' par le nom de votre interface réseau externe, comme 'eth0'.
ext_if="eth0"  # Remplacez par votre interface réseau externe
whitelist_file="/usr/local/bin/whitelist.txt"  # Chemin vers le fichier de liste blanche

# Vérifier que le fichier de liste blanche existe et est accessible en lecture
if [ ! -r "$whitelist_file" ]; then
    echo "Erreur : Le fichier de liste blanche '$whitelist_file' n'existe pas ou n'est pas
accessible en lecture." >&2
    exit 1
fi

# Lire le fichier de liste blanche et ajouter des règles pour chaque adresse IP ou plage CIDR
while read -r line; do
```

```
    # Ignorer les lignes de commentaire et les lignes vides
    if [[ ! "$line" =~ ^# ]] && [[ -n "$line" ]]; then
        # Détecter si l'adresse est IPv4 ou IPv6
        if grep -qE ':' <<< "$line"; then
            # Adresse IPv6
            ip6tables -I DOCKER-USER -i "$ext_if" ! -s "$line" -p tcp -m multiport --dports
80,443 -j DROP
            ip6tables -I DOCKER-USER -i "$ext_if" ! -s "$line" -p udp -m multiport --dports
80,443 -j DROP
        else
            # Adresse IPv4
            iptables -I DOCKER-USER -i "$ext_if" ! -s "$line" -p tcp -m multiport --dports
80,443 -j DROP
            iptables -I DOCKER-USER -i "$ext_if" ! -s "$line" -p udp -m multiport --dports
80,443 -j DROP
        fi
    fi
done < "$whitelist_file"
```

Crée la whitlist (pour l'exemple du proxy Cloudflare) :

```
sudo nano /usr/local/bin/whitelist.txt
```

```
# Cloudflare https://www.cloudflare.com/fr-fr/ips/
173.245.48.0/20
103.21.244.0/22
103.22.200.0/22
103.31.4.0/22
141.101.64.0/18
108.162.192.0/18
190.93.240.0/20
188.114.96.0/20
197.234.240.0/22
198.41.128.0/17
162.158.0.0/15
104.16.0.0/13
104.24.0.0/14
172.64.0.0/13
131.0.72.0/22
2400:cb00::/32
```

```
2606:4700::/32
2803:f800::/32
2405:b500::/32
2405:8100::/32
2a06:98c0::/29
2c0f:f248::/32
```

Créer un service systemd pour exécuter ce script avant Docker :

```
sudo nano /etc/systemd/system/iptables-rules.service
```

Contenu du fichier :

```
[Unit]
Description=Apply custom iptables rules
After=docker.service

[Service]
Type=oneshot
ExecStart=/usr/local/bin/iptables-rules.sh
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

Activer le service pour qu'il s'exécute au démarrage avant Docker :

```
sudo systemctl daemon-reload
sudo systemctl enable iptables-rules.service
```

(Re)démarrez le service système du pare-feu et Docker :

```
sudo systemctl restart iptables-rules.service
sudo systemctl restart docker
```

Pour tester la bonne application des règles, redémarrez le serveur puis vérifiez les logs des services pour vous assurer que tout fonctionne correctement :

```
sudo reboot
```

Après le redémarrage, consultez les logs du service IPTables personnalisé et du service Docker :

```
sudo journalctl -u iptables-rules.service
sudo journalctl -u docker.service
```

Documentation

- https://www.n0tes.fr/2019/05/11/Docker-et-IPtables/
- https://docs.docker.com/network/iptables/#restrict-connections-to-the-docker-daemon
- https://docs.docker.com/v17.09/engine/userguide/networking/default_network/custom-docker0/

Révision #24
Créé 2025-04-02 19:31:00 CEST par Philippe Favre
Mis à jour 2025-10-15 17:09:56 CEST par Philippe Favre