Installer et configurer le parefeu

- À partir de Debian Buster (v10) :
  - Le parefeu traditionnel *Iptables* est remplacé par *Nftables*.
  - *Firewalld* est un outil de gestion du parefeu (une surcouche) qui sait utiliser *Nftables*.
- Commandes réseaux utiles :
  - `netstat -anpt | grep LISTEN` ou `ss -lntu` : pour voir les ports TCP ouverts sur la machine
  - '' ss -lnue '' : pour voir les ports UDP ouverts sur la machine.
  - `ip link show` : pour trouver les noms des interfaces réseaux de la machine (ex. : eth0, eth1, ...).
- Pour utiliser Nftables directement, préparer sa configuration comme indiqué ci-dessous mais ce n'est pas nécessaire si vous utilisez `Firewalld` :
  - Documentation des commandes de nft
  - Le fichier de configuration principal : `/etc/nftables.conf`
  - Installer Nftables :
    - Debian 12 : (v1.8.9 - le 2023-07-03) `apt install nftables`
    - Debian 10 : en version backports (v1.8.3 - le 2020-01-03) si nécessaire (par défaut sous Debian Buster) : `apt install -t buster-backports nftables`
  - Activer automatiquement Ntables au démarrage : `systemctl enable nftables`
  - Voir la liste des règles : `nft list ruleset`
  - Créer la table IP FILTER : `nft add table ip filter`
    - Y créer la chaine INPUT : `nft add chain ip filter INPUT`
    - Y créer la chaine OUTPUT : `nft add chain ip filter OUTPUT`

Installer et configurer Firewalld

- Ressources :
  - How To Install and Configure Firewalld on Debian 10 (Buster)
  - CentOS 7 : Utilisation et configuration de firewalld
  - Understanding Firewalld in Multi-Zone Configurations
- Installer le paquet : `apt install firewalld`
  - Nftables est disponible en v0.9.8 sous Debian 11 (le 2023-05-12) : `apt install nftables`
  - Installer Nftables en version backports (v0.9.3 - le 2021-03-26) : `apt install -t buster-backports nftables`
- Configurer : `vi /etc/firewalld/firewalld.conf`
  - **ATTENTION** : au 2021-03-26, le support de Nftables n'est pas pris en compte dans Docker. Il est donc conseillé de maintenir l'utilisation d'`iptables` sur le système et donc de l'utiliser avec Firewalld. Voir les explications détaillées.
  - Modifier les propriétés suivantes :

```
# Pour corriger le bug :
https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=914694
IndividualCalls=yes
# Forcer l'utilisation de Nftables quand il sera supporté par
Docker avec la valeur suivante :
```

```
# FirewallBackend=nftables
# En attendant le support de Nftables par Docker nous utilisons iptables :
FirewallBackend=iptables
```

    - Il est aussi nécessaire de mettre à jour l'alternative `iptables` avec la commande : `update-alternatives --config iptables` : choisir le mode *legacy*.
- **ATTENTION** : avant d'activer le pare-feu veiller à avoir autoriser l'accès SSH sur le nouveau port. Pour cela configurer la *zone public* comme indiqué ci-dessous dans la section "Configuration des zones".
    - En cas de problème de connexion SSH suite à l'activation du pare-feu, il est possible de se connecter au serveur via la console VNC disponible sur le Manager d'Ovh au niveau des instances ou via l'interface Open Stack Horizon (aussi accessible depuis le Manager d'OVH).
- Démarrer le pare-feu si vous avez configurer au préalable la *zone public* : `systemctl start firewalld`
- Activer automatiquement au démarrage de la machine le parefeu : `systemctl enable firewalld`
- Afficher le status du service *Firewalld* : `systemctl status firewalld`

Quelques commandes Firewalld

- Afficher la version de *Firewalld* : `firewall-cmd --version`
- Vérifier l'état du parefeu : `firewall-cmd --state`
- Voir toutes les zones actives : `firewall-cmd --get-active-zones`
- Voir toutes les règles actives d'une zone particulière : `firewall-cmd --zone=<ma-zone> --list-all`
- Lister toutes les règles actives de la zone par défaut : `firewall-cmd --list-all`
- Afficher la zone par défaut : `firewall-cmd --get-default-zone`
- Afficher les zones actives (= liées à une interface) : `firewall-cmd --get-active-zones`
- Activer une zone en l'associant à une interface : `firewall-cmd --zone=<zone-name> --change-interface=<interface-name>`
- Voir la zone d'une interface : `firewall-cmd --get-zone-of-interface=<interface-name>`
- Lister les services activables/désactivables : `firewall-cmd --get-services`
- Activer un service de manière permanente dans une zone : `firewall-cmd --permanent --zone="<zone-name>" --add-service="<service-name>"`
- Pour ajouter/supprimer un *port* ou *service* utiliser les options du même nom avec les prévixes : `--add-<port|service>` ou `--remove-<port|service>`
- Recharger les règles sur les zones (celles ajoutées avec `--permanent` sont sauver, les autres sont annulées) : `firewall-cmd --reload`

Configuration des zones

Zone "public"

Activer les services suivant pour la zone *public* :

- Au préalable, vous pouvez vous assurer que ***ens3*** est bien associé à **l'IPv4 public** du serveur à l'aide de la commande : '' ip a ''
- Commandes *Firewalld* à exécuter pour la *zone public* :

```
firewall-cmd --set-default-zone=public
firewall-cmd --zone=public --permanent --change-interface=ens3
#+-----------------------------------------------------------------+
# Spécifique web-srv
firewall-cmd --zone=public --permanent --add-port=<port-ssh-web>/tcp
firewall-cmd --zone=public --permanent --add-service={http,https}
firewall-cmd --zone=public --permanent --remove-service={dhcpv6-
client,ssh}
# 2020-08-27 - Activer "masquerade" si vous obtenez un "502 Bad
Gateway" par le Nginx des Dockers
firewall-cmd --zone=public --permanent --add-masquerade
#+-----------------------------------------------------------------+
# Spécifique db-srv
firewall-cmd --zone=public --permanent --add-port=<port-ssh-db>/tcp
firewall-cmd --zone=public --permanent --remove-service={dhcpv6-
client,http,https,ssh}
#+-----------------------------------------------------------------+
# Spécifique bkp-srv
firewall-cmd --zone=public --permanent --add-port=<port-ssh-bkp>/tcp
firewall-cmd --zone=public --permanent --add-service={http,https}
firewall-cmd --zone=public --permanent --remove-service={dhcpv6-
client,ssh}
# 2020-08-27 - Activer "masquerade" si vous obtenez un "502 Bad
Gateway" par le Nginx des Dockers
firewall-cmd --zone=public --permanent --add-masquerade
#+-----------------------------------------------------------------+
firewall-cmd --reload
# Afficher le résultat :
firewall-cmd --info-zone=public
```

Zone "internal"

Activer les services suivant pour la zone *internal* (VPN entre les 2 serveurs) :

- Au préalable, vous pouvez vous assurer que ***ens7*** est bien associé à **l'IPv4 du VPN (10.0.1.xxx)** du serveur à l'aide de la commande : '' ip a ''
- Commandes *Firewalld* à exécuter pour la *zone internal* :

```
firewall-cmd --zone=internal --permanent --change-interface=ens7
firewall-cmd --zone=internal --permanent --remove-service={dhcpv6-
client,mdns,ssh,samba-client}
#+-----------------------------------------------------------------+
# Spécifique web-srv
firewall-cmd --zone=internal --permanent --add-port=<port-ssh-web>/tcp
# InfluxDb (si encore présent sur web-srv, normalement devrait être
```

```
basculer sur bkp-srv)
# firewall-cmd --zone=internal --permanent --add-port=8086/tcp
# firewall-cmd --zone=internal --permanent --add-port=8089/udp
# Docker API
firewall-cmd --zone=internal --permanent --add-port=2376/tcp
#+-----------------------------------------------------------------+
# Spécifique db-srv
firewall-cmd --zone=internal --permanent --add-port=<port-ssh-db>/tcp
# Postgresql
firewall-cmd --zone=internal --permanent --add-service=postgresql
# Docker API
firewall-cmd --zone=internal --permanent --add-port=2376/tcp
#+-----------------------------------------------------------------+
# Spécifique bkp-srv
firewall-cmd --zone=internal --permanent --add-port=<port-ssh-bkp>/tcp
# InfluxDb
firewall-cmd --zone=internal --permanent --add-port=8086/tcp
firewall-cmd --zone=internal --permanent --add-port=8089/udp

#+-----------------------------------------------------------------+
firewall-cmd --reload
firewall-cmd --info-zone=internal
```

Zone "docker"

Zone spécifique pour *Docker* qui sera nommée *docker* :

- Au préalable, vous pouvez vous assurer que ***docker0*** est bien associé à **l'IPv4 172.17.0.1/16** du serveur à l'aide de la commande : '' ip a ''
- Commandes *Firewalld* à exécuter pour la *zone docker* :

```
# Commencé par créer le réseau Docker si ce n'est pas déjà fait !
# Pour vérifier la présence du réseau utiliser : 'docker network ls' et
le détail avec 'docker network inspect nginx-proxy'
docker network create --driver=bridge --subnet=172.18.5.0/24 --ip-
range=172.18.5.0/24 --gateway=172.18.5.1 nginx-proxy
# Zone docker
# Depuis la version v20.10.0 Docker créé automatiquement la zone
"docker" et y associe ses interfaces (docker0, br-...)
# firewall-cmd --permanent --new-zone=docker
# firewall-cmd --zone=docker --permanent --change-interface=docker0
firewall-cmd --zone=docker --permanent --add-source=172.18.5.0/24
# Permettre l'accès aux sites web public depuis un container (voir si
la règle ci-dessous fonctionne dans le temps)
firewall-cmd --zone=docker --permanent --add-rich-rule='rule
family=ipv4 source address=172.18.5.0/24 accept'
# Il est peut être nécessaire d'ajouter la règle suivante sur la zone
"docker" en plus/à la place de celle existant sur "public".
firewall-cmd --zone=docker --permanent --add-masquerade
```

```
# Le 10-04-2022, l'accès à l'IP privée 10.0.1.30 (ping 10.0.1.30)
fonctionnait sur l'hôte mais plus dans le docker Borgmatic. Pour
résoudre le problème, ajout des 2 règles suivantes:
firewall-cmd --permanent --zone=docker --add-source=10.0.1.0/24
firewall-cmd --permanent --zone=docker --add-rich-rule='rule
family=ipv4 source address=10.0.1.0/24 accept'

#+-----------------------------------------------------------------+
# Spécifique web-srv
# Accéder au status du Nginx installé sur l'hôte
firewall-cmd --zone=docker --permanent --add-port=9090/tcp
# Si nécessaire, ajouter d'autres port TCP comme  50081 (cms-adminer),
50080 (cms-nginx), 50084 (wiki-sinp-nginx) avec la commande suivante :
#firewall-cmd --zone=docker --permanent --add-port=<YOUR-PORT>/tcp

#+-----------------------------------------------------------------+
# Spécifique bkp-srv
# Accéder au status du Nginx installé sur l'hôte
firewall-cmd --zone=docker --permanent --add-port=9090/tcp

#+-----------------------------------------------------------------+
firewall-cmd --reload
firewall-cmd --info-zone=docker
# ATTENTION : penser à redémarrer le service Docker après : systemctl
restart docker
```

Problème : Firewalld, Nftables et Docker

- **Symptômes** : les containers présent dans un network Docker de type bridge utilisateur n'arrivent pas à se connecter entre eux via leur nom de domaine.
- Il semblerait que les problèmes d'accès aux noms de domaines rencontrés lors de l'utilisation de Docker soient lié au fait que Docker ne supportent pas encore Nftables [2020-01-03].
  - ⇒ nous rétrogradons le système pour utiliser Iptables :
    - `update-alternatives --config iptables` : choisir le mode *legacy*
    - Arrêter *Nftables*, *Docker* et *Firewalld* : `systemctl stop firewalld ; systemctl stop nftables ; systemctl disable nftables ; systemctl stop docker`
    - Éditer `vi /etc/firewalld/firewalld.conf` et remettre :

      ```
      FirewallBackend=iptables
      ```

    - Supprimer toutes les règles permanentes : `rm -fR /etc/firewalld/zones/`
    - Redémarrer *Docker* puis *Firewalld* : `systemctl start firewalld ; systemctl start docker`
    - Recréer toutes les règles *Firewalld*
- La solution précédente semble être une solution retenue sur le ticket "[feature request] nftables support" de Mobi (la version en marque blanche de Docker). C'est peut être la plus simple à retenir pour l'instant. La solution consistant à utiliser une table ip et ipv6 à la place d'inet semble engendrer d'autres problèmes...

- Des ressources utiles sur le sujet :
  - Reddit : Docker, Debian Buster & nftables (2019-07-07) : résumé de la situation avec des liens au 2019-07-07.
  - IPv6 on production Docker (2017-06-05) : solution pour Debian utilisant Nftables mais pas Firewald... Peut être trop complexe à maintenir car nécessite une bonne connaissance du fonctionnement des pare-feux.
  - Github : [feature request] nftables support (#26824) : ticket du support de Nftables par Docker.
  - ServerFault: No network connectivity to/from Docker CE container on CentOS 8 : exemples de règles pour Firewalld permettant de faire fonctionner Docker correctement. NON TESTÉ...

Problème : Firewalld et Docker - "connect: no route to host"

- **Contexte** : Par défaut, *Firewalld* bloque la communication entre les containers sur le serveur hôte hébergeant Docker. Du coup, les containers n'arrive pas à accéder aux serveurs web public extérieur au container.
- **Vérification** : ce message d'erreur peut aussi apparaître si le service distant ne fonctionne pas. Assurez vous que le service distant est bien disponible sur le port attendu. Pour cela, utiliser `telnet` avec : `telnet <ip> <port>`. Si le service est bien accessible, vous pouvez appliquer les solutions suivantes.
- **Solutions** : pour autoriser la communication entres le containers *Docker*, il est possible d'utiliser une des solutions suivantes :
  - **Solution 1** (voir plutôt la solution 2 qui est pérenne) : `firewall-cmd --permanent --direct --add-rule ipv4 filter INPUT 4 -i docker0 -j ACCEPT`
    - Elle devra être suivi des commandes suivantes :
      - `firewall-cmd --reload`
      - `systemctl restart docker`
  - **Solution 2** : *Docker* créé de nouvelles interfaces réseau associés à un nouveau intervalle d'IP privés pour chaque nouveau réseau créé, visible via : `ip a`. Ainsi, associer l'interface *docker0* à une zone ne suffit pas, il semble plus pertinent de créer une zone spécifique et d'y associer en *source* les IP du réseau *Docker* que nous créons. A noter que depuis sa version 20.10.0, Docker créé automatiquement dans Firewalld la zone "docker" dans laquelle il associe les interfaces qu'il créé. Procédure :
    - Créer notre réseau Docker utilisateur, ici nommé *nginx-proxy*, de type *bridge* en spécifiant les IP possible pour les containers (`--subnet=172.18.5.0/24`) et la référence à l'IP de l'hôte dans ce réseau (`--gateway=172.18.5.1`) : `docker network create --driver=bridge --subnet=172.18.5.0/24 --gateway=172.18.5.1 nginx-proxy`
    - Créer une zone spécifique pour *Docker* dans *Firewalld* : `firewall-cmd --permanent --new-zone=docker`
    - Associer l'interface *docker0* à cette zone : `firewall-cmd --permanent --zone=docker --change-interface=docker0`
    - Associer le réseau Docker "utilisateur" que nous avons créé : `firewall-cmd --permanent --zone=docker --add-source=172.18.5.0/24`
      - Cette alternative semble mieux fonctionner : `firewall-cmd --zone=docker --permanent --add-rich-rule='rule family=ipv4 source address=172.18.5.0/24 accept'` : fonctionne correctement au 2021-03-26 avec la version 20.10.5 de Docker et Firewalld 0.6.3.
    - Activer sur la zone "public" la *masquerade* semble être nécessaire (à confirmer) :

```
firewall-cmd --zone=public --permanent --add-masquerade
```

    - Recharger pour supprimer les règles temporaires et appliquer les règles permanentes : `firewall-cmd --reload`
    - Redémarrer *Firewalld* : `systemctl restart firewalld`
    - Redémarrer *Docker* : `systemctl restart docker`
- **Notes** : stopper le service Firewalld puis redémarrer le service Docker et enfin démarrer à nouveau Firewalld résout "temporairement" le problème *connect: no route to host*. Il est nécessaire de faire cette manip à chaque redémarrage de container et sur chaque serveur avec lequel le container discute… ! En outre, cet ordre de démarrage des services provoque un autre type d'erreur (cf ci-dessous "Problème : iptables: No chain/target/match by that name").

Problème : "iptables: No chain/target/match by that name"

- **Erreur** :

```
ERROR: for manager-portainer  Cannot start service manager-portainer:
driver failed programming external connectivity on endpoint
0f0a6417f223_manager-portainer
(55e5c1439a5f2d8880f8e65676b492263223a2ea649aac87d72c4a0a98aac21c):
(iptables failed: iptables --wait -t nat -A DOCKER -p tcp -d 127.0.0.1
--dport 9000 -j DNAT --to-destination 172.18.5.2:9000 ! -i br-
e8fc729a5e58: iptables: No chain/target/match by that name.
```

    - **Contexte** : lors du démarrage du Docker Portainer (ex. manager.silene.eu) avec la commande : `docker-compose up`
    - **Solution** : Stopper le service Firewalld et Docker, puis redémarrer dans l'ordre Firewalld puis Docker résout "temporairement" le problème.

- **Erreur** :

```
mars 26 02:00:00 sinp-paca-web dockerd[8322]:
time="2021-03-26T02:00:00.644589374+01:00" level=info msg="ignoring
event"
container=c6e9cd33b6eb6f60edec8e645541de8d20afb15a7d911197c9bc89ab7a49e
780 module=libcontain
mars 26 02:00:00 sinp-paca-web dockerd[8322]:
time="2021-03-26T02:00:00.880242322+01:00" level=info msg="ignoring
event"
container=0c0f81af4cc08fa364adee66ee9202e7c28b883056e26db51d7b144a35f26
ccc module=libcontain
mars 26 02:00:02 sinp-paca-web dockerd[8322]:
time="2021-03-26T02:00:02.873481139+01:00" level=info msg="ignoring
event"
container=3a2ce9ad82ea0c1299ebce75e84707dc562a21eb9b2cb05696a1b4807042e
6cb module=libcontain
mars 26 02:00:03 sinp-paca-web dockerd[8322]:
time="2021-03-26T02:00:03.105072289+01:00" level=info msg="ignoring
event"
container=e15f443ce2a865ae783689bf103990bdf9804ac533d5772e2c14fbf6f102d
354 module=libcontain
```

```
mars 26 02:00:24 sinp-paca-web dockerd[8322]:
time="2021-03-26T02:00:24.468115748+01:00" level=warning msg="Failed to
allocate and map port 50081-50081:  (iptables failed: iptables --wait -
t nat -A DOCKER -p tcp -
mars 26 02:00:24 sinp-paca-web dockerd[8322]:
time="2021-03-26T02:00:24.544748120+01:00" level=error
msg="e15f443ce2a865ae783689bf103990bdf9804ac533d5772e2c14fbf6f102d354
cleanup: failed to delete container from c
mars 26 02:00:24 sinp-paca-web dockerd[8322]:
time="2021-03-26T02:00:24.544792461+01:00" level=error msg="Handler for
POST /v1.24/containers/cms-adminer/start returned error: driver failed
programming external con
mars 26 02:00:24 sinp-paca-web dockerd[8322]:
time="2021-03-26T02:00:24+01:00" level=info msg="Firewalld: docker zone
already exists, returning"
mars 26 02:00:25 sinp-paca-web dockerd[8322]:
time="2021-03-26T02:00:25+01:00" level=info msg="Firewalld: docker zone
already exists, returning"
mars 26 02:00:25 sinp-paca-web dockerd[8322]:
time="2021-03-26T02:00:25.711870702+01:00" level=warning msg="Failed to
allocate and map port 50080-50080:  (iptables failed: iptables --wait -
t nat -A DOCKER -p tcp -
mars 26 02:00:25 sinp-paca-web dockerd[8322]:
time="2021-03-26T02:00:25.784483347+01:00" level=error
msg="c6e9cd33b6eb6f60edec8e645541de8d20afb15a7d911197c9bc89ab7a49e780
cleanup: failed to delete container from c
mars 26 02:00:25 sinp-paca-web dockerd[8322]:
time="2021-03-26T02:00:25.784525338+01:00" level=error msg="Handler for
POST /v1.24/containers/cms-nginx/start returned error: driver failed
programming external conne
```

- **Contexte** : Les container cms-wordpress et wiki-sinp-nginx ne veulent pas démarrer et affiche une erreur lié à Iptables. Docker en version 20.10.0+ ajoute automatiquement une nouvelle zone *docker* à Firewalld comme indiqué dans la doc concernant Iptables de Docker. Il semblerait aussi que le serveur à 2h et 3h du matin effectue une tâche qui provoque l'erreur ci-dessous car *DOCKER* n'existe pas dans Iptables. Cette entrée disparait si on démarre le service Firewalld après celui de Docker. La commande '' iptables -t nat -nvL | grep DOCKER '' ne retourne rien alors qu'elle devrait affiché ceci :

```
2118 93277 DOCKER     all  --  *      *       0.0.0.0/0
0.0.0.0/0           ADDRTYPE match dst-type LOCAL
    0     0 DOCKER     all  --  *      *       0.0.0.0/0
!127.0.0.0/8         ADDRTYPE match dst-type LOCAL
```

- **Solution** : redémarrer le service Docker après celui de Firewalld. Le redémarrage du service Docker va recréer l'entrée *DOCKER*. Relancer ensuite les containers qui n'ont pas voulu démarré depuis l'interface de Portainer par exemple.

Problème : "psql: error: could not connect to server: Connection refused"

- **Contexte** : Depuis le container Borgmatic de l'instance *db-srv*, il est impossible d'accéder à Postgresql. La commande suivante `psql -h 172.18.0.1 -p 5432 -U postgres --no-password` génère l'erreur :

```
psql: error: could not connect to server: Connection refused
    Is the server running on host "172.18.5.1" and accepting
    TCP/IP connections on port 5432?
```

- **Solution** : sur l'hôte, afficher les IPs sur lesquelles Postgresql écoute avec la commande `ss -lntu` et vérifier qu'il y ait bien une ligne :

```
tcp        LISTEN      0           128
172.18.0.1:5432                  0.0.0.0:*
```

  Il faut Postgresql (port 5432) écoute sur l'IP de la gateway du bridge de Docker (ici 172.18.0.1). Si cette ligne est absente, redémarrer Postgresql `systemctl restart postgresql`. Elle devrait apparaitre. Si pour une raison ou une autre le service Docker n'est pas démarré au moment du démarrage de Postgresql (ordre du démarrage des services par la machine lors de son lancement), Postgresql n'écoutera pas sur l'IP 172.18.0.1 car non existante...

Problème : aucun accès à internet depuis le container

- ** Erreur** :

```
# Curl
Info: Could not resolve host: acme-v02.api.letsencrypt.org
# Ping
bad address 'www.google.com'
```

- **Contexte** : aucun accès à internet que cela soit via une IP (8.8.8.8) ou un nom de domaine (www.google.com) mais l'hôte via la passerelle (172.18.5.1) du reste joignable (`ping 172.18.5.1`).
- **Solution** : après avoir essayé les différentes techniques exposées sur cette page StackOveflow, c'est le redémarrage de la machine qui a tout remis dans l'ordre : `reboot`.

From:
https://sinp-wiki.cbn-alpin.fr/ - **CBNA SINP**

Permanent link:
https://sinp-wiki.cbn-alpin.fr/serveurs/installation/parefeu?rev=1688398616

Last update: **2023/07/03 15:36**