Installation de pare-feu redondants avec OpenBSD

Matthieu Herrb
Laboratoire d'Analyse et d'Architecture des Systèmes
7, avenue du Colonel Roche
BP 54200
31031 Toulouse cedex 4

1 Introduction

Ce tutoriel présente l'installation et la configuration d'un pare-feu utilisant le filtre de paquets PF sous OpenBSD.

OpenBSD est un système d'exploitation multi-plateformes (il tourne sur plusieurs architectures matérielles) de la famille Unix, dérivé de 4.4-BSD. Comme tous les systèmes de la famille BSD, il se distingue de Linux par son noyau spécifique, mais également par le fait que la même équipe développe le noyau et l'environnement en mode utilisateur. Les programmes tiers de l'environnement GNU/Linux sont disponibles via le mécanisme des *ports(5)*. Le projet OpenBSD produit une nouvelle version tous les six mois (en mai et en octobre généralement) et chaque version est supportée pendant un an. Les principales architectures matérielles supportées sont les processeurs de la famille Intel/AMD (en 64 et 32 bits), les ARM (v7 – 32 bits et v8 – 64 bits) les PowerPC (macppc et power9) et les SPARC 64 bits.

Outre le système, le projet OpenBSD est à l'origine de plusieurs projets annexes comme OpenSSH, LibreSSL, OpenBGPd, RPKI-client ou encore OpenSMTPd

2 Avant de commencer

2.1 Des architectures de pare-feu

Il n'est pas question de refaire ici un cours complet sur les architectures possibles pour mettre en place un pare-feu. Dans ce support, c'est le cas d'un pare-feu «extérieur» qui dispose principalement de deux interfaces, `LAN` et `WAN` et met potentiellement en œuvre de la traduction d'adresses (NAT) pour l'accès au WAN en IPv4 qui est étudié.

Cette configuration peut être adaptée facilement dans le cas d'un pare-feu interne, connectant plusieurs réseaux en étoile.

Une particularité de l'architecture présentée dans ce tutoriel est l'utilisation de domaines de routage (*rdomain(4)*) pour séparer l'interface d'administration (plan de contrôle du pare-feu) des interfaces actives (plan de données).

Dans le cas d'une configuration redondante, cette architecture permet au pare-feu passif d'accéder à internet pour se mettre à jour (ou pour des opérations de maintenance / configuration à distance) en traversant le pare-feu actif et en étant donc protégé comme tout autre noeud du réseau.

C'est en particulier très utile dans le cas où on ne dispose que d'une seule adresse IP routable coté WAN. Mais cela améliore aussi la sécurité et la facilité d'administration dans d'autres configurations.

2.2 Choix du matériel

Avant de commencer, il faut s'assurer d'avoir le matériel adapté et télécharger les médias d'installation.

OpenBSD supporte de nombreuses configurations matérielles. Mais il est important de s'assurer (via des tests en plus de la lecture des notes de distribution et des docs) que le matériel retenu est effectivement bien supporté.

Dans le cas de *machines physiques* on préférera :

- un processeur x86 64 bits (architecture **amd64**). Les versions actuelles d'OpenBSD ne sont pas capables d'exploiter pleinement le mode multiprocesseur. Il est donc préférable de choisir un processeur avec peu de cœurs mais une fréquence maximale de fonctionnement et/ou un cache de grande taille.

  Pour des pare-feu plus économes en énergie, on peut envisager des solutions sur architectures ARM ou MIPS 64 bits (arm64 ou octeon) mais le choix des machines disponibles est limité et leur support dans OpenBSD n'est pas toujours parfait. Il est préférable de procéder à des tests poussés avant de passer en production une telle solution.

- 8Go de RAM. Dans l'application présentée ici, la consommation de RAM est limitée. Il n'est pas nécessaire d'avoir plus de 16 Go de RAM. Des configurations avec 2 ou 4 Go de RAM sont envisageables également.

- un disque de 100Go. Il n'est pas nécessaire d'avoir beaucoup de stockage, sauf à vouloir conserver des volumes importants de traces de trafic.

- des cartes réseau de qualité. Il est conseillé d'éviter les cartes utilisant les puces RealTek (pilote *re(4)*).

Pour une machine virtuelle, les hyperviseurs basés sur kvm ou VMware fonctionnent. Le dimensionnement sera similaire à celui des machines physiques. On choisira des cartes réseau virtuelles de type *vio(4)* de préférence sur KVM et de type *vmx(4)* sur VMware.

Selon l'architecture retenue, une carte séparée par réseau à gérer sera utilisée, ou bien on pourra utiliser des VLANs sur une seule carte.

Pour une configuration redondante, deux machines identiques sont préférables à une configuration trop hétérogène.

Il faut prévoir un lien dédié pour la synchronisation entre les deux machines (pfsync) et une interface d'administration séparée.

Sur une machine physique avec une architecture de type WAN/LAN, quatre interfaces réseau sont donc nécessaires.

2.3 Description de la configuration du tutoriel

Ce tutoriel repose sur une machine physique, sur laquelle est installé OpenBSD, et trois machines virtuelles installées via *vmm(4)*, l'hyperviseur d'OpenBSD.

Il s'agit ici de produire une configuration de test facilement reproductible.

En particulier la machine hôte dans ce tutoriel facilite la simulation de l'accès au réseau internet externe et joue en même temps le rôle de machine / réseau dédié à l'administration des pare-feu. Dans un déploiement en production il faudrait bien sûr séparer ces deux rôles.

Voici les machines virtuelles qui sont utilisées :

- routeur-a
- routeur-b
- client : une machine dans le LAN pour tests

Sur les deux routeurs il y a quatre interfaces réseau (virtuelles) principales (qui sont des interfaces physiques lorsque le routeur est une machine physique) :

- vio0 / vio1 → réseaux à protéger
- vio2 → pfsync
- vio3 → admin

Trois réseaux virtuels utilisant les interfaces *veb(4)* sont définis, pour interconnecter les interfaces ci-dessus :

- veb0 : correspondant au WAN (connexion vers l'internet), ici connectée au monde extérieur sur une interface physique de l'hôte.

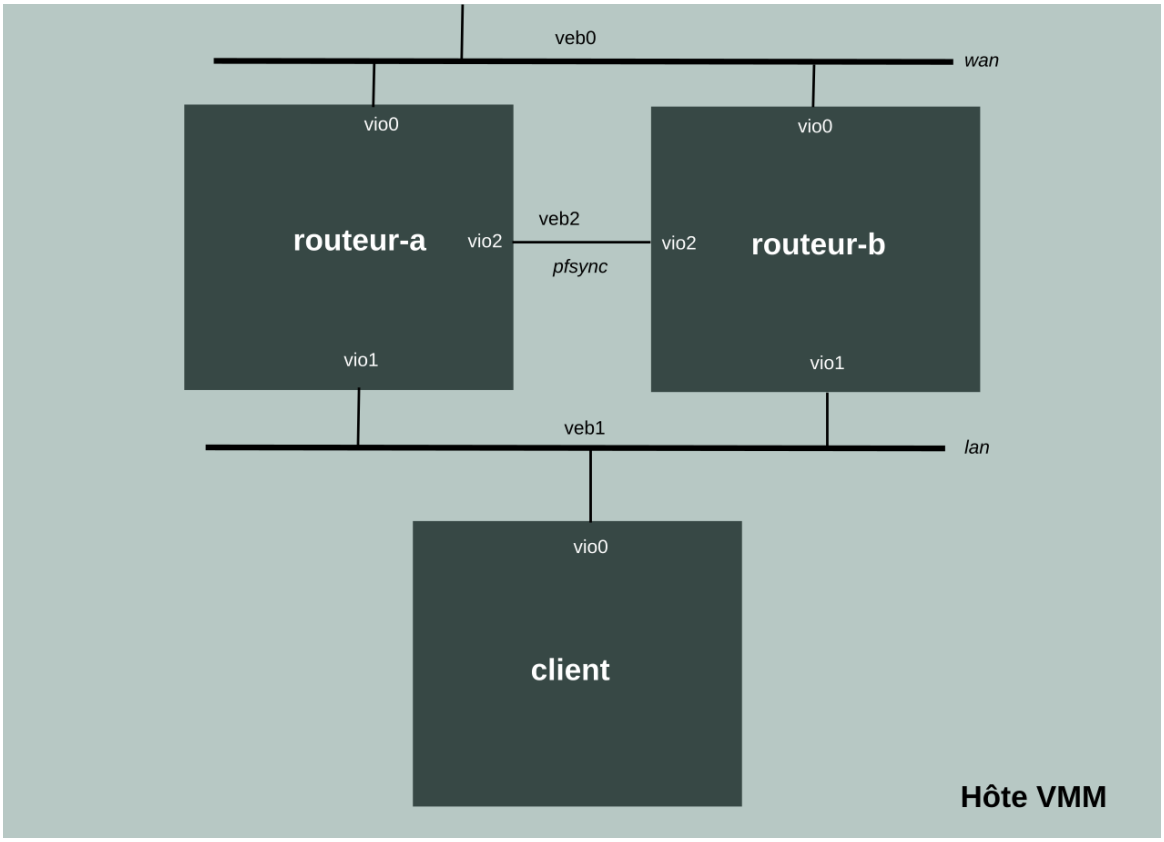

Figure 1 - Architecture du tutoriel

- veb1 : correspondant au LAN (machines protégées par les pare-feux)
- veb2 : correspondant au lien de synchronisation entre les pare-feux (*pfsync*)

Dans ce tutoriel l'hôte dispose de deux interfaces physiques : l'une est utilisée pour l'administration de la plateforme de test, et l'autre est connectée à «l'extérieur» (accès à internet, par exemple via un VPN ou un routeur 4G). Si l'on dispose d'une troisième interface physique, il est possible de la connecter à un LAN physique qui sera alors protégé. Dans cette présentation on n'utilise pas cette possibilité.

3 Intallation initiale

L'image d'installation pour un démarrage sur clé USB de l'installeur OpenBSD pour amd64 est ici

3.1 Procédure d'installation d'OpenBSD

L'installation initiale se résume aux étapes suivantes :

1. Démarrage sur le média d'installation (clé USB…) ;
2. Dans cette image système minimale :
3. Configuration réseau ;
4. Partitionnement du disque système ;
5. Paramètres système : nom de la machine et du domaine, compte root, …

6. Initialisation du disque système pour permettre de démarrer dessus
7. Reboot sur le disque système ;
8. Finalisation de la configuration :
9. Installation de paquets supplémentaires ;
10. Configuration finale des interfaces réseau, des services et des règles de filtrage.

Durant l'installation initiale, la configuration du réseau va se limiter à une seule interface, avec le strict minimum pour permettre un accès à internet afin de télécharger les paquets logiciels qui seront nécessaires durant la configuration.

La configuration complète sera faite dans un second temps.

3.2 Configuration du réseau

Sous OpenBSD, la configuration réseau se fait via les fichiers `/etc/hostname.*` (voir *hostname.if(5)*) pour les différentes interfaces.

Avant de commencer, il faut donc identifier toutes les interfaces à configurer. Le noyau nomme les interfaces réseau à partir du nom du pilote correspondant au matériel. Ainsi une interface réseau Intel 1Gb/s sera gérée par le pilote *em(4)* et donc la première instance de ce pilote créera l'interface `em0`, la seconde `em1` etc. La table ci-dessous indique les pilotes les plus fréquemment rencontrés.

Fabricant	type de carte	pilote
Broadcom	1Gb/s	*bge(4)*
Intel	1Gb/s	*em(4)*
Intel	X5x0 10Gb/s	*ix(4)*
Intel	700 series 10Gb/s	*ixl(4)*
Realtek	1Gb/s	*re(4)*
Realtek	2.5Gb/s	*rge(4)*
VirtIO	interface virtuelle	*vio(4)*
VMWare	interface virtuelle	*vmx(4)*

Il est possible d'ajouter un champ `description` lors de la configuration d'une interface afin de pouvoir s'y retrouver. Par contre il n'est pas possible de renommer des interfaces (comme le permet par exemple Linux).

Pour créer un pare-feu dans une configuration redondante quatre interfaces au moins seront nécessaires :

- deux interfaces connectées aux deux réseaux entre lesquels le pare-feu sera situé (ou plus si on souhaite gérer plus de deux réseaux)
- une interface dédiée à la synchronisation de l'état entre les pare-feu ; il est vraiment recommandé d'utiliser une interface dédiée, ce trafic doit être sécurisé et le plus robuste possible
- une interface dédiée à l'administration des pare-feu. Celle-ci est optionnelle, mais cela offre à la fois une meilleure sécurité et cela simplifie les opérations de maintenance.

Idéalement, pour une redondance complète, chaque interface vers les réseaux sera un agrégat de deux interfaces physiques, vers 2 commutateurs indépendants. Cependant une seule interface suffit bien souvent à condition que chaque pare-feu soit connecté à un commutateur séparé.

Chaque interface est configurée par un fichier `/etc/hostname.`**if** où **if** est l'interface à configurer. Ce fichier utilise un pseudo-langage de configuration, très proche des arguments de la commande shell *ifconfig(8)*, mais avec quelques spécificités.

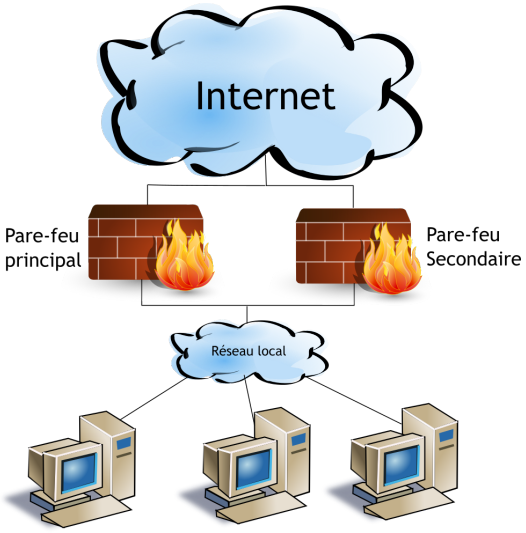

Figure 2 - Exemple d'architecture

4 Concepts réseau d'OpenBSD

4.1 Domaines de routage

OpenBSD supporte plusieurs domaines de routage, qui permettent interfaces réseau indépendantes les unes des autres. Chaque table de routage est incluse dans un domaine de routage, qui est un espace d'adresses IP séparé dans le noyau. Par défaut toutes les tables de routage sont dans le domaine 0 (le domaine par défaut). Il suffit de déclarer un domaine de routage différent lors de la configuration d'une interface réseau pour la placer dans ce domaine. Normalement, le routage des adresses IP d'un domaine de routage reste à l'intérieur de ce domaine, mais des règles de *pf(4)* peuvent être utilisées pour router des paquets d'un domaine vers un autre.

Chaque processus du système utilise une table de routage (et est donc attaché à un domaine de routage). Il est possible d'indiquer dans la configuration des processus serveurs au démarrage du système à quelle table de routage chacun d'entre eux est attaché. Cela permet ainsi par exemple d'avoir un démon SSH attaché uniquement à une interface dédiée à l'administration du système. Sans avoir à modifier le fichier *sshd_config(8)*, celui-ci ne sera visible qu'à partir du réseau d'administration en question. Toute tentative de connexion sur le port 22 (SSH) depuis une autre interface, échouera, car dans son domaine de routage aucun processus n'est en écoute sur ce port.

La figure 3 présente un exemple avec deux domaines de routage. Dans le domaine `0`, il n'y qu'une seule interface réseau : `em2`. Les processus `proc3`, `proc4` et `proc5` appartenant à ce domaine ne voient que cette interface. Le domaine `1` contient les interfaces `em0` et `em1`. Si le routage (*ip_forwarding*) des paquets est autorisé, ceux-ci pourront passer entre ces deux interfaces (mais pas vers `em2`). Les processus `proc1` et `proc2` ne voient que ces deux interfaces.

Dans ce tutoriel, deux domaines de routage sont utilisés : un domaine dédié correspondra au plan de données d'un routeur : il ne voit que les interfaces des réseaux qu'il inter-connecte et ne fait que router les paquets entre ces interfaces. Il n'est pas nécessaire de pouvoir se connecter au routeur depuis une interface de ce domaine particulier. Les seuls services actifs dans ce domaine seront ceux uilisés par le réseau : DHCP, Router Advertisements, NTP ou DNS.

Le domaine par défaut est celui de l'administration du routeur, utilisé pour se connecter via SSH sur les machines qui le composent, pour faire les mises à jour logicielles ou les modification de la configuration. Il correspond au plan de contrôle d'un routeur. Contrairement aux interfaces du domaine utilisé pour le routage, les interfaces du domaine d'administration ne seront pas virtualisées par CARP et auront donc chacune une adresse IP propre, afin de pouvoir gérer individuellement chaque routeur.

Le choix de placer l'interface d'administration dans le domaine par défaut correspond à une simplification pratique : c'est dans ce domaine que l'administration système se fait, et l'interface réseau configurée lors de l'installation initiale dans le domaine par défaut reste l'interface d'administration par la suite (ici c'est `vio3`).

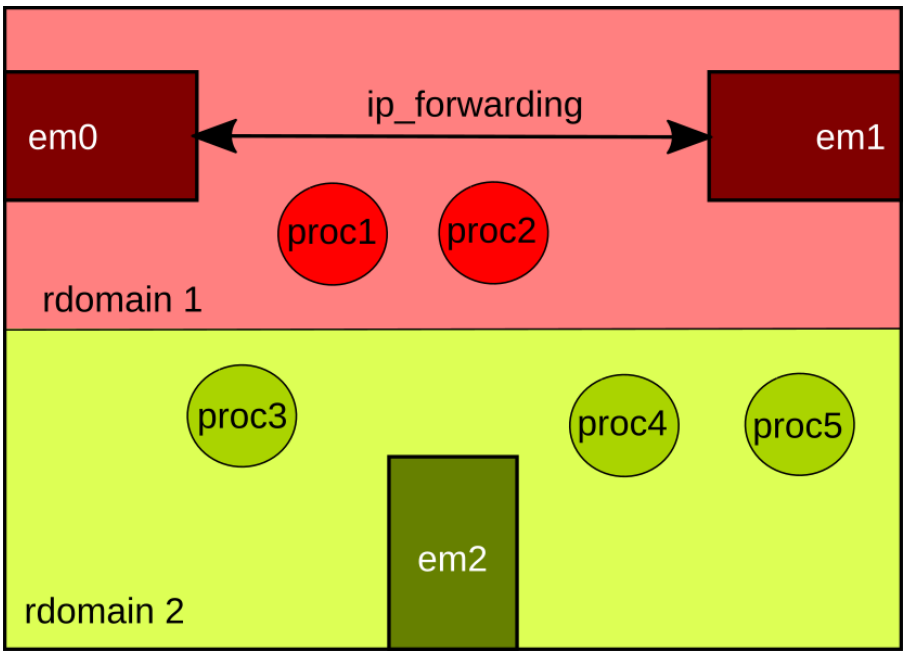

Figure 3 - Exemple de domaines de routage

4.2 Filtrage de paquets avec PF

PF (Packet Filter) est un sous-système de filtrage de paquets IP (version 4 ou 6), intégré à OpenBSD [1], où il a remplacé un autre filtre *ipfilter* qui a été retiré pour des raisons de licence. PF se place en entrée et en sortie des interfaces réseau. Un fichier de configuration décrit les actions qui peuvent être réalisées sur les paquets qui traversent le filtre. Ces actions peuvent être :

- passer le paquet tel quel
- modifier le paquet (NAT ou redirections)
- marquer le paquet (la marque pourra être utilisée plus loin)
- ou enfin bloquer le paquet (en générant éventuellement une réponse ICMP *permission denied* ou un RST dans le cas du protocole TCP

La syntaxe complète des règles de PF est décrite dans la page de manuel *pf.conf(4)*.

4.3 CARP et pfsync

CARP est un protocole qui permet de réaliser des routeurs redondants virtuels, comparable au protocole VRRP de Cisco. Il permet à plusieurs hôtes de partager une adresse IP via une adresse MAC virtuelle. Dans ce tutoriel CARP est utilisé dans le mode actif/passif : un seul routeur physique est actif à la fois. Lorsque le routeur passif détecte qu'il n'y a plus de routeur actif, il devient actif.

Chaque interface *carp(4)* a besoin d'un identifiant et un d'un secret partagé par ses membres. Lors du déploiement d'un réseau complexe où plusieurs interfaces virtuelles CARP sont présentes (plusieurs routeurs, ou bien d'autres services redondés), il est utile de consigner l'ensemble des identifiants (et éventuellement des secrets associés) de manière centrale pour éviter les conflits.

Le filtrage de paquets par PF étant un mécanisme qui gère l'état des connexions, il est nécessaire de synchroniser cet état (représenté par la **table des états**) entre les routeurs. C'est le rôle du protocole *pfsync(4)* qui assure la synchronisation.

Pour plus de robustesse, pfsync utilise dans ce tutoriel une connexion ethernet directe entre les deux pare-feux physiques à synchroniser. Il est possible d'envisager des configurations où pfsync utilise un autre réseau, en ajoutant une couche de réseau privé virtuel (IPsec par exemple) pour sécuriser ce trafic, mais cela sort du cadre de ce tutoriel.

4.4 VLANS

Il est possible d'utiliser les configurations présentées dans ce tutoriel pour gérer du trafic entre réseaux virtuels IEEE 802.1Q (vlans).

Sous OpenBSD, la configuration de vlans se fait en deux étapes : la configuration de l'interface physique d'abord, puis celle des interfaces correspondantes aux vlan.

La configuration d'une interface de type «trunk» supportant plusieurs vlans se fait en déclarant simplement l'interface active (`up`)

Une interface fille (qui reçoit et émet les paquets tagués avec un vlan particulier) se déclare en spécifiant l'interface parente et le numéro du vlan : **`vnetid`** *`n`* **`parent`** *`ifname`*

5 Administration du système

Cette section présente quelques outils d'administration système spécifiques à OpenBSD.

5.1 Ajout de paquets

L'installation des logiciels tiers se fait par le mécanisme des *ports(5)* qui permet de compiler ces logiciels de manière adaptée à OpenBSD. Mais heureusement, pour installer un logiciel (et ses dépendances) il n'est pas nécessaire de le compiler soi-même. OpenBSD fourni des *paquets binaires* (*packages(7)*) pré-compilés pour tous les logiciels de l'arbre des ports qui autorisent une telle distribution. L'installation d'un paquet se fait à l'aide de la commande *pkg_add(1)*.

Parmi les paquets intéressants, on retrouve les interpréteurs de commandes les plus connus (`bash`, `zsh`,...) les éditeurs de texte plus complets que les outils de base (`emacs`, `vim`,...) ainsi que les environnements de bureau et tous leurs outils (Gnome, KDE, XFCE,...)

5.2 Mises à jour de sécurité

En cas de problème de sécurité découvert après la diffusion d'une version, OpenBSD produit des correctifs de sécurité sous forme binaire directement installables via la commande *syspatch(8)*.

Chaque version d'OpenBSD est maintenue pendant un an (c'est à dire qu'il y deux versions maintenues : la version courante et la version immédiatement précédente).

Pour les paquets tiers, seule la dernière version est supportée par des mises à jour de sécurité. Pour mettre à jour les paquets suite à un avis de sécurité, la commande `pkg_add -u` est utilisée (voir *pkg_add(1)*)

5.3 Mise à jour du système

Les mises à jour majeures du système (tous les six mois) peuvent se faire à l'aide de la commande *sysupgrade(8)*. Celle-ci va :

- télécharger la distribution binaire de la nouvelle version
- redémarrer le système dans l'installeur avec un fichier de réponse automatique pour piloter la mise à jour sans intervention humaine
- lorsque la mise à jour est terminée, l'installeur redémarre le nouveau système.

En cas de problème pendant la procédure tout est fait pour permettre de redémarrer sur l'ancien système plutôt que de bloquer la procédure. Cela permet d'utiliser cette commande pour faire des mises à jour de systèmes distants en limitant les risques d'en perdre l'accès en cas de problème.

Après la mise à jour initiale, il reste à installer d'éventuels correctifs de sécurité (publiés depuis la sortie de la version majeure) et à mettre à jour les paquets installés.

Enfin, la commande *sysmerge(8)* permet de fusionner les modifications locales des fichiers de configuration de `/etc` de manière interactive. Lorsqu'une nouvelle version majeure apporte des modifications dans un fichier de configuration, si celui-ci n'avait pas été modifié, la nouvelle version sera installée automatiquement. Si le fichier avait été modifié localement, `sysmerge` permet de visualiser les différences entre les versions et de fusionner interactivement les modifications locales dans la nouvelle version. Il est également possible d'ignorer complètement la nouvelle version ou encore d'écraser complètement les modifications locales.

`sysmerge` peut être interrompu à tout moment. Tant que tous les fichiers de configuration affectés n'ont pas été traités, il suffit de le relancer pour reprendre au premier fichier qui n'a pas encore été fusionné.

Pour récapituler, voici les étapes pour passer à la version majeure suivante :

```
sysupgrade -r
syspatch
sysmerge
pkg_add -u
```

6 Configuration réseau du pare-feu redondant

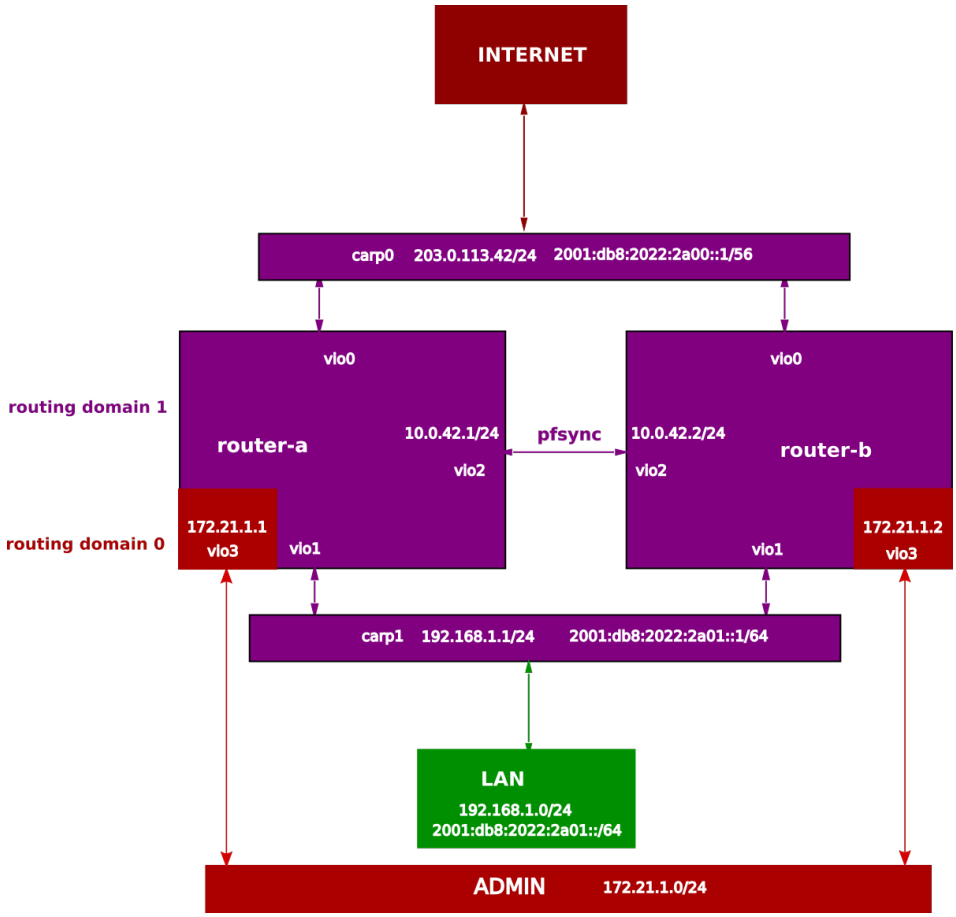

Figure 4 - Configuration réseau du tutoriel

L'installation va se faire à partir d'un réseau existant, si possible fonctionnel avec un accès internet. Les interfaces d'administration seront configurées dans ce réseau et l'accès internet sera utilisé pour les mises à jour éventuelles de paquets.

Dans le cas où l'on monte une toute nouvelle infrastructure, ce réseau peut être un réseau temporaire (partage de connexion 4G sur un téléphone,...). Il devra être reconfiguré une fois le réseau d'administration définitif en place.

Par contre dans le cas où les nouveaux pare-feux sont amenés à remplacer une solution existante, il peut tout à fait s'agir du réseau d'administration de la solution existante.

6.1 Interface d'administration

L'interface d'administration va utiliser le domaine de routage **0**, soit le domaine par défaut.

Pour simplifier la configuration, seul IPv4 est configuré sur le réseau d'administration en mode client DHCP. Il est bien sûr possible (et même recommandé) de configurer aussi (ou seulement) IPv6 sur ce réseau.

Sur les 2 routeurs :

`/etc/hostname.vio3`

```
inet autoconf
```

6.2 Configuration système

Sur les deux routeurs, le routage (*forwarding*) est activé pour IPv4 et IPv6, et la préemption de CARP est activée :

`/etc/sysctl.conf`

```
net.inet.ip.forwarding=1
net.inet6.ip6.forwarding=1
net.inet.carp.preempt=1
```

6.3 Interfaces CARP

Les interfaces coté internet (`vio0`) et coté LAN (`vio1`) sont activées sur les deux routeurs :

`/etc/hostname.vio0` et `/etc/hostname.vio1`

```
up
rdomain 1
```

Ensuite les interfaces CARP sont configurées. La seule différence entre les 2 routeurs est que sur le routeur secondaire (`router-b`), le paramètre `advskew` est défini avec une valeur de 200, afin que les annonces CARP de ce routeur ne soient envoyées que si aucune annonce n'est reçue du routeur principal 200ms après le moment où elle était attendue.

routeur A `/etc/hostname.carp0`

```
rdomain 1
vhid 10 carpdev vio0 pass SuperSecret10
inet 203.0.113.42 255.255.255.255
inet6 2001:db8:2022:2a00::1/56
! route -T1 add default 203.0.113.1
! route -T1 add -inet6 default fe80::1%carp0
```

routeur B `/etc/hostname.carp0`

```
rdomain 1
vhid 10 advskew 200 carpdev vio0 pass SuperSecret10
inet 203.0.113.42 255.255.255.255
inet6 2001:db8:2022:2a00::1/56
! route -T1 add default 203.0.113.1
! route -T1 add -inet6 default fe80::1%carp0
```

routeur A `/etc/hostname.carp1`

```
rdomain 1
vhid 20 carpdev vio1 pass SuperSecret20
inet 192.168.1.1 255.255.255.0
inet6 2001:db8:2022:2a01::1/64
```

routeur B `/etc/hostname.carp1`

```
rdomain 1
vhid 20 advskew 200 carpdev vio1 pass SuperSecret20
inet 192.168.1.1 255.255.255.0
inet6 2001:db8:2022:2a01::1/64
```

6.4 pfsync

routeur A `/etc/hostname.vio2`

```
rdomain 1
up
inet 10.0.42.1 255.255.255.0
```

routeur B `/etc/hostname.vio2`

```
rdomain 1
up
inet 10.0.42.2 255.255.255.0
```

les 2 routeurs `/etc/hostname.pfsync0`

```
rdomain 1
up syncdev vio2
```

7 Filtrage avec pf

Seules les fonctions de base de PF sont présentées dans ce tutoriel. Le jeu de règles proposé implémente un routeur d'accès à internet avec du NAT pour IPv4 et un réseau /64 pour IPv6 coté interne. La politique de filtrage laisse sortir le trafic et bloque le trafic venant de l'internet.

7.1 Interfaces utilisées

Avec PF et CARP, le filtrage ne se fait pas sur l'interface carp elle même, mais sur l'interface sous-jacente. Dans le cas de ce tutoriel, il s'agira des interfaces `vio`.

Par ailleurs PF utilise les notions de *in* et *out* pour indiquer les directions dans lesquelles les paquets traversent une interface. Lors de l'écriture d'une politique de filtrage, il faut faire attention à ce que ces directions correspondent à l'idée que l'on se fait de l'intérieur et de l'extérieur du réseau.

Dans le cas de l'accès internet, en plaçant les règles sur l'interface coté internet (l'interface **externe**), la direction `in` de pf correspondra bien au trafic entrant sur le réseau local et réciproquement la direction `out` correspondra bien au trafic sortant.

Dans le cas d'un pare-feu interne, les notions de *entrant* et *sortant* sont d'avantage relatives, il faut donc faire preuve d'attention pour placer les règles sur l'interface où cela sera le plus pertinent (*c-à-d* pas trop ambigu).

7.2 Accès internet

L'interface `carp0` sur `vio0` est l'interface vers l'internet et l'interface `carp1` sur `vio1` est l'interface coté LAN. Les règles seront donc placées sur l'interface **`vio0`**. Dans le cas simple considéré ici, l'interface `vio1` peut être ignorée par PF.

La traduction d'adresse (NAT) est activée depuis le LAN vers l'extérieur pour IPv4.

En IPv6, un sous-réseau de taille `/64` du préfixe `/56` est utilisé sur le LAN.

Les règles de filtrage minimales présentées ici laissent passer tout le trafic sortant et ne laissent entrer aucun trafic.

`/etc/pf.conf`

```
ext=carp0
ext_if=vio0
int=carp1
int_if=vio1
sync=pfsync0
sync_if=vio2
adm=vio3

set skip on { lo $int_if $adm }

set loginterface $ext_if

block log

# Laisse passer pfsync
pass quick on $sync_if proto pfsync \
    keep state (no-sync)
pass quick on $sync_if proto icmp

# Laisse passer carp
pass quick on $ext_if proto carp \
    keep state (no-sync)

pass out on $ext_if inet nat-to ($ext)
pass out on $ext_if inet6
```

Lors d'une modification du fichier `pf.conf`, la commande *pfctl(8)* est utilisée pour charger la nouvelle version en mémoire.

```
  pfctl -n -f /etc/pf.conf
```

L'option `-n` permet de faire l'analyse syntaxique du fichier de configuration sans appliquer les nouvelles règles. En cas d'erreur de syntaxe, une erreur sera affichée sans risque de laisser le système dans un état incohérent.

Une fois la syntaxe validée, la commande :

```
pfctl -f /etc/pf.conf
```

modifie réellement la configuration du filtre.

Il faut noter qu'en cas d'erreur de syntaxe dans le nouveau fichier de configuration, la configuration courante n'est en général pas modifiée, même si l'option `-n` n'est pas spécifiée. Cependant c'est une bonne pratique de toujours valider la syntaxe avant de charger les nouvelles règles.

Dans le cas de pare-feu redondants, il faut également propager la configuration au pare-feu inactif : copier le nouveau fichier de configuration et appliquer la configuration. Un script tel que celui proposé ci-dessous permet d'automatiser cela.

```
#! /bin/ksh
autre="routeur-b"
if /sbin/pfctl -n -f /etc/pf.conf; then
      echo "syntaxe ok -> update"
      pfctl -f /etc/pf.conf
else
      echo "erreur dans pf.conf"
      exit 2
fi
echo "mise a jour $autre"
scp -p /etc/pf.conf $autre:/etc/
ssh autre /sbin/pfctl -f /etc/pf.conf
exit 0
```

8 Autres services

Les autres services utiles pour un routeur d'accès internet seront installés à l'identique sur les deux pare-feux. Il s'agit de fournir les services de base de configuration des adresses IPv4 et IPv6 et de la résolution des noms DNS pour les machines du réseau local.

8.1 Serveur DCHP IPv4

dhcpd(8) distribue les adresses IPv4 pour le réseau local.

Le fichier de configuration est `/etc/dhcpd.conf` :

```
server-name "router-a";

subnet 192.168.1.1 netmask 255.255.255.0 {
      option routers 192.168.1.1;
      option subnet-mask 255.255.255.0;
      option domain-name-servers 192.168.1.1;
      range 192.168.1.100 192.168.1.254;
      allow bootp;
}
```

Deux serveurs DHCP sur une configuration redondante peuvent être synchronisés à l'aide des options `-y` et `-Y` pour diffuser et recevoir les mises à jour des baux DHCP. Dans ce tutoriel les interfaces `vio2` sont utilisées pour transporter ces mises à jour en multicast (similaire à `pfsync`).

Activation et démarrage du service :

```
# rcctl enable dhcpd
# rcctl set dhcpd rtable 1
# rcctl set dhcpd flags "-y vio2 -Y vio2 carp1"
# rcctl start dhcpd
```

8.2 Auto-configuration IPv6

rad(8) assure l'auto-configuration IPv6 des clients. Il est configuré par le fichier `/etc/rad.conf`

```
interface carp1 {
      prefix 2a03:7220:8085:2a01::/64
      dns {
            nameserver 2a03:7220:8085:2a01::1
            search my.domain
      }
}
```

Il n'y a pas d'état dans rad, donc il n'y a pas à considérer la synchronisation entre les deux routeurs.

Activation et démarrage du service :

```
# rcctl enable rad
# rcctl set rad rtable 1
# rcctl start rad
```

8.3 Résolveur DNS

unbound(8) est le serveur cache DNS récursif pour le LAN. Son fichier de configuration est situé dans sa cage *chroot(2)* : `/var/unbound/etc/unbond.conf` :

```
server:
      interface: 192.168.1.1
      interface: ::

      access-control: 0.0.0.0/0 refuse
      access-control: 127.0.0.0/8 allow
      access-control: 192.168.1.0/24 allow
      access-control: ::0/0 refuse
      access-control: ::1 allow
      access-control: 2a03:7220:8085:2a01::/64 allow
      access-control: fe80::/16 allow
```

Pas de besoin de synchronisation non plus.

Activation et démarrage du service :

```
# rcctl enable unbound
# rcctl set unbound rtable 1
# rcctl start unbound
```

9 Traces et statistiques

Plusieurs mécanismes sont disponibles pour observer l'état des pare-feux et caractériser leur activité sur le trafic qui les traverse.

9.1 Traces du pare-feu

Pour la collecte des traces du pare-feu, l'approche de PF est de rediriger les paquets marqués pour être tracés vers une interface réseau virtuelle dédiée : *pflog(4)*. Les outils standard de capture de traces réseau tels *tcpdump(8)* peuvent être utilisés pour examiner ces traces, tout en bénéficiant de la puissance du langage de filtrage de PCAP (*pcap-filter(5)*).

9.2 Statistiques

L'outil *systat(1)* permet d'observer en temps-réel un ensemble de statistiques sur l'état du système. Il a propose plusieurs vues sur le système. Parmi elles, deux sont intéressantes pour suivre l'activité du pare-feu :

- `rules` qui liste les règles actives et les statistiques associées (paquets et octets traités par la règle,...)
- `states` qui liste les états actifs du pare-feu, chaque état correspondant à un flux réseau. Pour chaque état les statistiques (nombre de paquets et d'octet traités, âge,...) sont affichées.

Voici un exemple d'affichage par la commande `systat rules`, affichant l'état des règles du pare-feu du tutoriel après quelques heures d'activité :

```
     1 users Load 0.00 0.00 0.00                       router-a.my.domain 18:38:51

RULE  ACTION    DIR LOG Q IF      PR        K     PKTS    BYTES   STATES   MAX INFO
   0  Block     Any Log                         3756   467746        0       drop all
   1  Pass      Any     Q vio2    pfsync    K     8161  1546728       53       all
   2  Pass      Any     Q vio2    icmp      K        0        0        0       all
   3  Pass      Any       vio2    udp       K       50     6400       44       all
   4  Pass      Any     Q vio0    carp      K    23284  1536744        2       all
   5  Pass      Out       vio0              K   667187  610111K      596       inet all
   6  Pass      In        vio0    icmp      K        0        0        0       all
   7  Pass      In        vio0    ipv6-icmp K     1528   114080      616       all
   8  Pass      Out       vio0              K     8070  5330068      864       inet6 all
```

Dans les ports on trouve également l'outil pftop qui fourni des informations similaires à `systat states` :

```
pfTop: Up State 1-26/26, View: default, Order: bytes, Cache: 10000                                                  18:47:11

PR        DIR SRC                       DEST                            STATE              AGE       EXP     PKTS    BYTES
carp      Out fe80::200:5eff:fe00:10a[ ff02::12[0]                   SINGLE:NO_TRAFFIC   06:34:47  00:00:29    11891   903716
carp      Out 192.168.31.210:0         224.0.0.18:0                  SINGLE:NO_TRAFFIC   06:34:47  00:00:29    11891   665896
pfsync    Out 10.0.42.1:0              10.0.42.2:0                 MULTIPLE:MULTIPLE     00:00:35  00:00:59       11     7596
icmp      Out 192.168.31.210:17756     194.57.3.79:8                      0:0            00:00:09  00:00:09       18     1512
udp       Out 192.168.31.210:62941     192.33.4.12:53              MULTIPLE:SINGLE       00:00:12  00:00:18        2     1117
udp       Out 2a03:7220:8081:6101::d2[ 2001:500:f::1[53]           MULTIPLE:SINGLE       00:00:12  00:00:18        2     1023
udp       Out 2a03:7220:8081:6101::d2[ 2001:678:c::1[53]           MULTIPLE:SINGLE       00:00:12  00:00:18        2      861
udp       Out 192.168.31.210:50372     193.176.144.22:53           MULTIPLE:SINGLE       00:00:12  00:00:18        2      812
udp       Out 2a03:7220:8081:6101::d2[ 2001:660:3001:4002::9[53    MULTIPLE:SINGLE       00:00:09  00:00:21        2      709
udp       Out 192.168.31.210:50702     193.176.144.22:53           MULTIPLE:SINGLE       00:00:09  00:00:21        2      689
udp       Out 192.168.31.210:53698     199.19.53.1:53              MULTIPLE:SINGLE       00:00:13  00:00:18        2      677
udp       Out 192.168.31.210:56589     193.49.159.9:53             MULTIPLE:SINGLE       00:00:09  00:00:21        2      670
udp       Out 2a03:7220:8081:6101::d2[ 2001:660:500a:2210::10[5    MULTIPLE:SINGLE       00:00:12  00:00:18        2      435
udp       Out 192.168.31.210:61181     193.52.36.172:53            MULTIPLE:SINGLE       00:00:12  00:00:18        2      407
udp       Out 192.168.31.210:62599     193.55.90.12:53             MULTIPLE:SINGLE       00:00:12  00:00:18        2      395
udp       Out 10.0.42.1:8067           10.0.42.2:8067                SINGLE:NO_TRAFFIC   00:00:37  00:00:00        3      384
udp       Out 2a03:7220:8081:6101::d2[ 2001:638:d:b103::1[53]      MULTIPLE:SINGLE       00:00:12  00:00:18        2      353
udp       Out 2a03:7220:8081:6101::d2[ 2001:638:d:b103::1[53]      MULTIPLE:SINGLE       00:00:12  00:00:18        2      353
udp       Out 192.168.31.210:52395     193.174.75.58:53            MULTIPLE:SINGLE       00:00:12  00:00:18        2      313
udp       Out 192.168.31.210:56066     193.174.75.58:53            MULTIPLE:SINGLE       00:00:12  00:00:18        2      313
udp       Out 192.168.31.210:61498     193.174.75.58:53            MULTIPLE:SINGLE       00:00:12  00:00:18        2      313
udp       Out 192.168.31.210:55054     193.49.159.2:53             MULTIPLE:SINGLE       00:00:12  00:00:18        2      191
ipv6-icmp In  2a03:7220:8081:6101::200 2a03:7220:8081:6101::d2[  NO_TRAFFIC:NO_TRAFFIC    00:00:08  00:00:02        2      136
ipv6-icmp Out 2a03:7220:8081:6101::d2[ 2a03:7220:8081:6101::200  NO_TRAFFIC:NO_TRAFFIC    00:00:03  00:00:07        2      136
ipv6-icmp Out fe80::200:5eff:fe00:10a[ fe80::31:200[135]         NO_TRAFFIC:NO_TRAFFIC    00:00:03  00:00:07        2      136
udp       In  10.0.42.1:8067           10.0.42.2:8067            NO_TRAFFIC:SINGLE       00:00:37  00:00:00        0        0
```

Pour l'intégration dans une solution de supervision plus globale, OpenBSD dispose d'un serveur SNMP (*snmpd(8)*) qui met à disposition, en plus des traditionnelles MIB présentant les statistiques par interface, des informations spécifiques aux actions de PF (cause de blocage des paquets, nombre d'états, etc.)

Pour une analyse fine du trafic, le noyau peut également exporter des informations sur les flux traversant le pare-feu au format Netflow. Pour cela il est possible de configurer une interface *pflow(4)* qui exportera des trames UDP contenant les enregistrement netflow sélectionnés. Plusieurs outils dans les ports permettent d'exploiter ces informations.

Enfin pour une analyse détaillée au niveau applicatif, les ports de type *divert(4)* permettent de recevoir au niveau utilisateur une copie de paquets bruts reçus et sélectionnés par pf et, après traitement, de le réinjecter (tels quels ou modifiés) dans la pile réseau. Ce mécanisme permet d'implémenter des outils d'analyse protocolaire, pour, par exemple intercepter certaines requêtes DNS ou certains URLs.

10 Annexe : Machines virtuelles sous OpenBSD

Ce tutoriel a été réalisé à l'aide de machines virtuelles sur un système OpenBSD doté de deux interfaces physiques.

10.1 Configuration réseau de l'hôte

Les deux interfaces physiques sont configurées ainsi :

em0 est connectée au routeur internet du site. Comme ce sont les VM **router-a** et **router-b** qui vont se connecter, cette interface est membre d'un commutateur virtuel *veb(4)*. Elle n'a pas d'adresse IP affectée.

em1 est connectée au réseau d'administration du site. Ce réseau dispose d'un serveur DHCP et permet aux machines de sortir si nécessaire vers internet (la configuration exacte n'est pas décrite ici, mais elle n'a pas d'importance). C'est par ce réseau que se fait l'administration des pare-feu.

`/etc/hostname.em0`

```
up
```

`/etc/hostname.em1`

```
inet autoconf
```

Trois commutateurs virtuels sont utilisés.

veb0 connexion internet, l'interface `em0` de l'hôte est incluse dans ce commutateur ainsi que les interfaces `vio0` de chaque routeur.

veb1 LAN virtuel. Connecte les interfaces `vio1` de chaque routeur ainsi que l'interface `vio0` de la machine virtuelle **client**

veb2 lien pfsync entre les deux routeurs. Connecte les interfaces `vio2` des deux routeurs.

`/etc/hostname.veb0`

```
up
add em0
```

`/etc/hostname.veb1` et `/etc/hostname.veb2`

```
up
```

Enfin, chaque machine virtuelle allant utiliser quatre interfaces virtuelles, il est nécessaire de créer des entrées supplémentaires de type `tap` dans `/dev` via *MAKEDEV(8)* :

```
# cd /dev
# for i in `jot 6 4`; do
# ./MAKEDEV tap$i
# done
```

10.2 Configuration des machines virtuelles

Trois machines virtuelles sont configurées sur l'hôte : les deux routeurs dont la configuration a été décrite en détails dans ce tutoriel et une machine **client** de test connectée au LAN virtuel, afin de vérifier que les routeurs fonctionnent.

Dans l'exemple les images disque des machines sont dans le dossier `/local/vm` et c'est l'utilisateur `admin` qui sera autorisé à les gérer :

Le fichier `/etc/vm.conf` (*vm.conf(5)*) qui décrit les machines virtuelles est créé en premier :

```
switch ext {
      enable
      interface veb0
}

switch int {
      enable
      interface veb1
}

switch pfsync {
      enable
      interface veb2
}

vm "router-a" {
  disable
  memory 1G
  disk "/local/vm/a.img"
  cdrom "/local/vm/install71.iso"
  boot device cdrom
  interface {
    switch "ext"
  }
  interface {
    switch "int"
  }
  interface {
    switch "pfsync"
  }
  local interface
  owner admin
}

vm "router-b" {
   disable
   memory 1G
   disk "/local/vm/b.img"
   cdrom "/local/vm/install71.iso"
   boot device cdrom
   interface {
      switch "ext"
   }
   interface {
     switch "int"
   }
   interface {
      switch "pfsync"
   }
```

```
    local interface
    owner admin
}

vm "client" {
    disable
    memory 1G
    disk "/local/vm/c.img"
    cdrom "/local/vm/install71.iso"
    boot device cdrom
    interface {
      switch "int"
    }
    owner admin
}
```

Les images disque sont initialisées avec *vmctl(8)* :

```
# mkdir -p /local/vm
# vmctl create -s 20G /local/vm/a.img
# vmctl create -s 20G /local/vm/b.img
# vmctl create -s 20G /local/vm/c.img
```

Puis l'image ISO d'installation est téléchargée :

```
# cd /local/vm
# ftp https://cdn.openbsd.org/pub/OpenBSD/7.1/amd64/install71.iso
```

Ensuite le service *vmd(8)* peut être activé et démarré :

```
# rcctl enable vmd
# rcctl start vmd
```

10.3 Installation des machines virtuelles

1. Installation système

   Pour l'installation du système démarrer la première VM avec *vmctl(8)* :

   ```
   # vmctl start -c routeur-a
   ```

   Pour l'installation des deux routeurs, seule l'interface `vio3` est configurée, en mode auto-configuration IPv4.

   Tout le partitionnement disque peut se faire en acceptant la configuration par défaut.

   À la fin de l'installation, il faut arrêter (*power down*) la machine virtuelle, pour modifier le fichier de configuation `vm.conf` afin de mette en commentaire les deux lignes qui permettaient le démarrage sur l'image d'un CD-ROM.

   Avant de re-lancer la machine virtuelle il faut informer le démon *vmd(8)* de la modification de la configuration :

   ```
   # rcctl reload vmd
   # vmcctl start -c routeur-a
   ```

2. Configuration

   Une fois connecté avec le compte root et le mot de passe choisi lors de l'installation, le routeur peut être configuré comme indiqué dans la section 6.

À la fin de l'installation de `router-a`, le deuxième routeur `routeur-b` est installé et configuré de la même façon.

Enfin, lorsque les deux pare-feu sont installés, il est possible de passer à l'installation de la machine virtuelle `client`, située sur le réseau local qui servira aux tests.

Pour celle-ci il n'y a qu'une seule interface réseau (`vio0`) à configurer et elle sera laissée en mode autoconfiguration en IPv4 et en IPv6.

Bibliographie

[1] P. N. M. Hansteen. *Book of PF, 3rd Edition*. No Starch Press, 2014. https://nostarch.com/pf3.

[2] X. Cartron. *Héberger son serveur avec OpenBSD, 3e édition*. Atramenta, 2017. https://si3t.ch/ah/.

[3] M. Herrb. Outils de sécurité réseau dans un laboratoire avec Openbsd et PF. Dans *Actes du congrès JRES 2011*, Toulouse, Novembre 2011. https://homepages.laas.fr/matthieu/talks/2011/jres-2011-pf.pdf.

[4] M. Herrb. PF - Packet Filter. Dans *ANF CNRS Sécurité informatique pour administrateurs systèmes et réseaux (SIARSv2)*, Toulouse, Novembre 2018. https://homepages.laas.fr/matthieu/talks/siarsv2-pf-obsd.pdf.

[5] S. Rapenne. Full WireGuard setup with OpenBSD. Rapport technique, Dataswamp.org, Octobre 2021. https://dataswamp.org/~solene/2021-10-09-openbsd-wireguard-exit.html.

Table des matières