Ponts filtrants

Résumé

Souvent il est utile de diviser un réseau physique (comme un réseau Ethernet) en deux segments séparés sans avoir à créer des sous-réseaux, et utiliser de routeur pour les relier ensemble. Le dispositif qui connecte les deux réseaux de cette manière est appelé un pont. Un système FreeBSD avec deux cartes réseau est suffisant pour jouer le rôle d'un pont.

Un pont fonctionne en scannant les adresses au niveau MAC (adresses Ethernet) des cartes connectées à chacune de ses interfaces réseau et puis transfère le trafic entre les deux réseaux si la source et la destination sont situées sur un segment différent. Sous beaucoup de points de vue un pont est similaire à un commutateur (switch) Ethernet avec uniquement deux ports.

Version française de Marc Fonvieille <blackend@FreeBSD.org>.

Table des matières

1. Pourquoi utiliser un pont filtrant?

De plus en plus fréquemment, grâce à la baisse des coûts des connexions Internet haut débit (xDSL) et aussi en raison de la réduction des adresses IPv4 disponibles, beaucoup de compagnies sont connectées à l'Internet 24 heures sur 24 et avec peu (parfois même pas une puissance de 2) d'adresses IP. Dans ces situations il est souvent désirable d'avoir un coupe-feu qui filtre le trafic entrant et sortant depuis et vers l'Internet, mais la solution d'un filtrage de paquet basé sur un routeur peut ne pas être applicable, soit en raison de problèmes de sous-réseau, le routeur appartient au fournisseur d'accès (ISP), ou parce qu'il ne supporte pas une telle fonction. Dans ces scénarios l'utilisation d'un pont filtrant est fortement conseillée.

Un coupe-feu basé sur un pont peut être configuré et inséré entre le routeur xDSL et votre concentrateur/commutateur Ethernet sans aucun problème d'adresse d'IP.

2. Comment l'installer

Ajouter les fonctions de pont à un système FreeBSD n'est pas difficile. Depuis la 4.5-RELEASE il est possible de charger de telles fonctionnalités sous forme de module au lieu d'avoir à recompiler le

noyau, simplifiant énormément la procédure. Dans les sous-sections suivantes j'expliquerai les deux méthodes d'installation.

Ne pas suivre les deux méthodes: une procédure *exclue* l'autre. Choisissez la meilleure en fonction de vos besoins et capacités.

Avant d'aller plus loin, soyez sûr de disposer deux cartes Ethernet qui supportent le mode promiscuous pour la réception et la transmission, comme elles doivent être capables d'envoyer des paquets Ethernet avec n'importe quelle adresse, et non pas juste pour la leur. De plus, pour avoir de bonnes performances, les cartes devront être capable contrôler le bus PCI (PCI bus mastering). Les meilleurs choix sont toujours l'Intel EtherExpress™ Pro, suivie de la série 3c9xx de chez 3Com®. Pour simplifier la configuration il peut être utile d'avoir deux cartes de différents constructeurs (utilisant un pilote de périphérique différent) afin de distinguer clairement quelle interface est connectée au routeur et quelle autre est connectée au réseau.

2.1. Configuration du noyau

Donc vous avez décidé d'utiliser la bonne vieille méthode d'installation. Pour commencer, vous devez ajouter les lignes suivantes à votre fichier de configuration du noyau:

```
options BRIDGE
options IPFIREWALL
options IPFIREWALL_VERBOSE
```

La première ligne est pour compiler le support du pont, la seconde est le coupe-feu et la troisième sont les fonctions de traces du coupe-feu.

Maintenant il est nécessaire de compiler et d'installer le nouveau noyau. Vous pourrez trouver des instructions détaillées dans la section Compiler et installer un noyau sur mesures du Manuel de FreeBSD.

2.2. Le chargement de modules

Si vous avez choisi d'employer cette méthode nouvelle et plus simple; la seule chose à faire maintenant est d'ajouter la ligne suivante au fichier /boot/loader.conf:

```
bridge_load="YES"
```

De cette façon, durant le démarrage du système le module bridge.ko sera chargé avec le noyau. Il n'est pas nécessaire de rajouter une ligne pour le module ipfw.ko, étant donné qu'il sera chargé automatiquement après l'exécution des étapes présentées dans la section suivante.

3. Dernière préparation

Avant de redémarrer afin de charger le nouveau noyau ou les modules nécessaires (selon la

méthode d'installation précédemment retenue), vous devez faire quelques modifications dans le fichier de configuration /etc/rc.conf. La règle de défaut du coupe-feu est de rejeter tous les paquets IP. Au départ nous configurerons un coupe-feu "ouvert", afin de vérifier son fonctionnement sans problème relatif au filtrage de paquet (dans le cas où vous faite cela à distance, une telle configuration vous évitera de rester isolé du réseau). Ajoutez les lignes suivantes dans /etc/rc.conf:

```
firewall_enable="YES"
firewall_type="open"
firewall_quiet="YES"
firewall_logging="YES"
```

La première ligne activera le coupe-feu (et chargera le module ipfw.ko s'il n'est pas compilé dans le noyau), la seconde le configurera dans le mode "ouvert" (comme expliqué dans /etc/rc.firewall), la troisième ligne rendra le chargement des règles silencieux (sans affichage) et la quatrième activera le support de trace d'activité du coupe-feu.

Au sujet de la configuration des interfaces réseau, la méthode la plus utilisée est d'assigner une adresse IP à une seule des cartes réseau, mais le pont fonctionnera à l'identique si les deux interfaces ou aucune n'ont d'adresse IP configurée. Dans le dernier cas (sans adresse IP) la machine faisant office de pont sera toujours cachée et inaccessible depuis le réseau: pour la configurer, vous devez vous attacher depuis la console ou à travers une troisième interface réseau séparée du pont. Parfois, durant le démarrage du système, certains programmes ont besoin d'accéder au réseau, par exemple pour la résolution de noms: dans ce cas il est nécessaire d'assigner une adresse IP à l'interface externe (celle connectée à l'Internet où le serveur DNS réside), puisque le pont sera activé à la fin de la procédure de démarrage. Cela signifie que l'interface fxp0 (dans notre cas) doit être mentionnée dans la section concernant ifconfig du fichier /etc/rc.conf, mais pas xl0. Assigner une adresse IP aux deux cartes réseau n'a pas beaucoup de sens, à moins que, durant la procédure de démarrage, des applications devront accéder à des services sur les deux segments Ethernet.

Il y a une autre chose importante à savoir. Quand on utilise l'IP sur Ethernet, il y a en fait deux protocoles Ethernet utilisés: l'un est l'IP, l'autre est l'`ARP`. ARP effectue la conversion de l'adresse IP d'un hôte en son adresse Ethernet (couche MAC). Afin d'autoriser la communication entre deux hôtes séparés par le pont, il est nécessaire que le pont transmette les paquets ARP. Un tel protocole n'est pas inclus dans la couche IP, puisque qu'il n'apparaît qu'avec l'utilisation de l'IP sur Ethernet. Le coupe-feu de FreeBSD filtre exclusivement la couche IP et donc tous les paquets non-IP (ARP compris) seront transmis sans être filtrés, même si le coupe-feu est configuré pour ne rien laisser passer.

Il est maintenant temps de redémarrer le système et de l'utiliser comme auparavant: il y aura de nouveaux messages concernant le pont et le coupe-feu, mais le pont ne sera pas activé et le coupe-feu, étant en mode "ouvert", n'interdira aucune opération.

Si il y a un quelconque problème, vous devriez le corriger maintenant avant de continuer.

4. Activer le pont

A ce point, pour activer le pont, vous devez exécuter les commandes suivantes (en pensant bien de

remplacer les noms des deux interfaces réseau fxp0 et xl0 avec les vôtres):

```
# sysctl net.link.ether.bridge.config=fxp0:0,xl0:0
# sysctl net.link.ether.bridge.ipfw=1
# sysctl net.link.ether.bridge.enable=1
```

La première ligne spécifie quelles interfaces devront être activées par le pont, la seconde activera le coupe-feu sur le pont et enfin la troisième activera le pont.

Si vous utiliser FreeBSD 5.1-RELEASE ou une version précédente, les variables sysctl diffèrent. Consultez la page de manuel bridge(4) pour plus de détails.

A ce moment là, vous devriez être en mesure d'insérer la machine entre deux ensembles d'hôtes sans compromettre de capacités de communication entre eux. Ensuite, l'étape suivante est d'ajouter les parties `net.link.ether.bridge.[bla]=[bla]` de ces lignes dans le fichier /etc/sysctl.conf afin de les exécuter au démarrage.

5. Configurer le coupe-feu

Il est maintenant temps de créer votre propre fichier de règle personnalisé pour le coupe-feu, afin de sécuriser le réseau interne. Il y aura quelques complications à faire cela parce que toutes les fonctionnalités du coupe-feu ne sont pas disponibles sur un pont. En outre, il y a une différence entre les paquets qui sont en cours de transmission et les paquets qui sont reçus par la machine locale. En général, les paquets entrants traversent le coupe-feu une seule fois, et pas deux comme c'est normalement le cas; en fait ils sont filtrés à la réception, aussi les règles qui utilisent "out" ou "xmit" n'agiront jamais. Personnellement, j'utilise "in via" qui est une ancienne syntaxe, mais qui a un sens quand vous la lisez. Une autre limitation est que vous êtes réduit à l'utilisation seulement des commandes "pass" ou "drop" pour les paquets filtrés par un pont. Les choses sophistiquées comme "divert", "forward" ou "reject" ne sont pas disponibles. De telles options peuvent toujours être utilisées, mais uniquement sur le trafic à destination ou depuis le pont lui-même (s'il possède une adresse IP).

Une nouveauté de FreeBSD 4.0, est le concept de filtrage avec gestion des états (stateful). C'est une grande amélioration pour le trafic `UDP`, qui typiquement est une requête de sortie, suivie peu de temps après par une réponse avec le même ensemble d'adresses IP et de numéro de ports (mais avec une source et une destination réservée, bien sûr). Pour les coupe-feux qui n'ont pas de gestion des états, il n'y a presque pas de possibilité de gérer ce type de trafic en une seule session. Mais avec un coupe-feu qui peut se "souvenir" d'un paquet `UDP` sortant et qui, pour quelques minutes, autorise une réponse, la gestion des services `UDP` est triviale. L'exemple suivant montre comment le faire. Il est possible de faire la même chose avec les paquets `TCP`. Cela vous permet d'éviter certaines attaques par refus de service et autres désagréables problèmes, mais augmente également rapidement la taille de votre table des états.

Regardons un exemple de configuration. Notez tout d'abord qu'au début du fichier /etc/rc.firewall il y a déjà des règles standards pour l'interface de boucle locale lo0, aussi nous ne devrions pas y faire attention. Les règles personnalisées devraient être placées dans un fichier séparé (disons /etc/rc.firewall.local) et chargées au démarrage du système, en modifiant la ligne de /etc/rc.conf où

nous avions défini le coupe-feu "ouvert":

```
firewall_type="/etc/rc.firewall.local"
```

Vous devez préciser le chemin *complet*, sinon il ne sera pas chargé avec le risque de rester isolé du réseau.

Pour notre exemple imaginez que nous avons l'interface fxp0 connectée vers l'extérieur (Internet) et l'interface xl0 vers l'intérieur (LAN). Le pont possède une adresse IP `1.2.3.4` (il n'est pas possible que votre fournisseur d'accès puisse vous donner une adresse de classe A comme celle-ci, mais pour cet exemple cela ira).

```
# Les choses dont nous avons gardé l'état avant de
continuer
add check-state

# Rejeter les réseaux RFC 1918
add drop all from 10.0.0.0/8 to any in via fxp0
add drop all from 172.16.0.0/12 to any in via fxp0
add drop all from 192.168.0.0/16 to any in via fxp0

# Autorise la machine pont à communiquer si elle le désire
# (si la machine est sans adresse IP, ne pas inclure ces lignes)
add pass tcp from 1.2.3.4 to any setup keep-state
add pass udp from 1.2.3.4 to any keep-state
add pass ip from 1.2.3.4 to any

# Autorise les hôtes internes à communiquer
add pass tcp from any to any in via xl0 setup keep-state
add pass udp from any to any in via xl0 keep-state
add pass ip from any to any in via xl0

# Section TCP
# Autoriser SSH
add pass tcp from any to any 22 in via fxp0 setup keep-state
# Autoriser SMTP seulement vers le serveur de courrier
add pass tcp from any to relay 25 in via fxp0 setup keep-state
# Autoriser le transfert de zone seulement par le serveur de noms esclave
[dns2.nic.it]
add pass tcp from 193.205.245.8 to ns 53 in via fxp0 setup keep-state
# Laisser passer les sondes d'ident.  C'est préférable plutôt que d'attendre
# qu'elles s'arrêtent
add pass tcp from any to any 113 in via fxp0 setup keep-state
# Laisser passer la zone de "quarantaine"
add pass tcp from any to any 49152-65535 in via fxp0 setup keep-state

# Section UDP
# Autorise le DNS seulement vers le serveur de noms
add pass udp from any to ns 53 in via fxp0 keep-state
# Laisser passer la zone de "quarantaine"
add pass udp from any to any 49152-65535 in via fxp0 keep-state

# Section ICMP
# Laisser passer 'ping'
add pass icmp from any to any icmptypes 8 keep-state
# Laisser passer les messages d'erreurs générés par 'traceroute'
add pass icmp from any to any icmptypes 3
add pass icmp from any to any icmptypes 11

# Tout le reste est suspect
add drop log all from any to any
```

Ceux qui parmi vous ont déjà installé des coupe-feux auparavant pourront noter l'absence de certaines choses. En particulier, il n'y a pas de règles contre le forgeage d'adresses, en fait nous n'avons *pas* ajouté:

```
add deny all from 1.2.3.4/8 to any in via fxp0
```

Cela, bloque les paquets provenant de l'extérieur et clamant être en provenance de votre réseau. C'est quelque chose que vous devriez habituellement faire pour être sûr que personne n'essaie de passer outre votre filtre de paquet, en générant d'infames paquets qui sont comme s'il venaient de l'intérieur. Le problème avec cela est qu'il y a *au moins* un hôte de l'extérieur que vous ne voulez pas ignorer: le routeur. Mais habituellement, les fournisseurs d'accès contrent le forgeage sur leur routeur, aussi nous ne devons pas nous en préoccuper plus que cela.

La dernière règle semble être une copie conforme de la règle par défaut, qui ne laisse passer rien de ce qui n'est pas spécifiquement autorisé. Mais il y a une différence: tout le trafic suspect sera tracé.

Il y a deux règles pour faire passer le trafic SMTP et DNS vers le serveur de courrier et le serveur de noms, si vous en avez. Evidemment l'ensemble du jeu de règle devra être arrangé selon vos goûts personnels, c'est juste un exemple particulier (le format des règles est décrit précisément dans la page de manuel ipfw(8)). Notez qu'afin que "relay" et "ns" puissent fonctionner, les services de résolution de nom doivent fonctionner *avant* que le pont soit activé. C'est un exemple pour être sûr que vous avez configuré l'adresse IP sur la bonne carte réseau. Alternativement il est possible de spécifier l'adresse IP au lieu du nom (nécessaire si la machine est sans adresse IP).

Les personnes qui ont l'habitude de configurer des coupe-feux ont probablement également utilisé soit une règle "reset" soit une règle "forward" pour les paquets ident (TCP port 113). Malheureusement, ce n'est pas une option applicable avec un pont, aussi la meilleure solution est de les laisser passer vers leur destination. Aussi longtemps que la machine de destination ne fait pas tourner un daemon ident, cela est relativement inoffensif. L'alternative est de bloquer les connexions sur le port 113, ce qui pose problème avec des services comme l'IRC (la sonde d'ident doit s'arrêter).

La seule autre chose qui est un peu étrange que vous avez peut-être noté est qu'il y a une règle pour laisser le pont communiquer, et une autre pour les hôtes internes. Rappelez-vous que c'est parce que les deux ensembles de trafic prendront un chemin différent à travers le noyau et dans le filtre de paquet. Le réseau interne ira à travers le pont, alors que la machine locale utilisera la pile IP normale pour communiquer. Ainsi les deux règles s'occupent de cas différents. La règle "in via fxp0" fonctionne pour les deux chemins. En général, si vous employez des règles "in via" dans tout le filtre, vous devrez faire une exception pour les paquets générés localement, parce qu'ils ne sont pas passés par une de nos interfaces.

6. Participants

Plusieurs parties de cet article proviennent, et furent mises à jour et adaptées d'un vieux texte sur les ponts, écrit par Nick Sayer. Cet article est également inspiré d'une introduction sur les ponts de Steve Peterson.

Un grand merci à Luigi Rizzo pour l'implémentation du code de pont dans FreeBSD et pour le temps qu'il a passé à répondre à toutes mes questions à ce sujet.

Un remerciement également à Tom Rhodes qui a supervisé mon travail de traduction de l'Italien (langue d'origine de cet article) à l'Anglais.