Pont + pare−feu + DSL Mini−HOWTO

Derek Ney

<derek@hipgraphics.com>

Geneviève Gracian – Traduction française

Xavier Venient – Relecture de la version française

Version originale du 9 novembre 2000

Version française du 25 novembre 2001

Configurer un système Linux de façon à ce qu'il se comporte comme un pont et un pare−feu avec une connexion DSL.

Table des matières

1. Introduction

1.1. L'histoire

L'écriture de ce document a débuté le 10 décembre 1999 par Derey Ney `<derek@hipgraphics.com>` après trois jours de frustration avec le pontage et le filtrage après avoir basculé d'un réseau PPP à un lien DSL.

1.2. Nouvelles versions

Les nouvelles versions peuvent êtres trouvées à www.tldp.org.

1.2.1. Historique des versions

v0.04 (9 novembre 2000)

- mise à jour pour le nouvel utilitaire de configuration de pont bridgex

v0.03 (24 mars 2000)

- correction de l'URL pour BRCFG.tgz

v0.02 (13 décembre 1999)

- intégration des corrections de Leonard Dickens (merci Leonard !)

v0.01 (10 décembre 1999)

- version initiale

1.3. Copyrights

2. Pontage, pares−feu et connexions DSL

Jusqu'il y a peu de temps, notre réseau était relié au réseau global via PPP par un modem. Avec cette configuration, j'avais installé un pare−feu utilisant IPChains et cela fonctionnait correctement. Nous avons récemment évolué vers une configuration DSL. Je pensais que ce serait une bagatelle de simplement basculer mon pare−feu de façon à ce qu'il m'isole du réseau entrant associé à la connexion DSL. J'avais tort. J'ai mis trois jours avant de parvenir à un résultat opérationnel. J'ai trouvé beaucoup d'informations douteuses sur le réseau qui m'ont procuré bien du désarroi. Ce mini−HOWTO a été écrit parce que je soupçonnais que notre installation serait plutôt banale dans le futur avec l'extansion de DSL et que je souhaitais aider les autres à s'épargner une importante frustration.

Je pense que ceci est applicable à une connexion modem−câble mais « c'est vous qui voyez » dans la mesure où je ne connais rien aux liaisons par modem−câble.

2.1. Le problème

Le problème que je tente de résoudre est de de configurer le système de façon à ce que le code du pare−feu dans le noyau (qui est manipulé par ipchains) soit utilisable pour filtrer les paquets qui vont et viennent entre le monde extérieur et le réseau local. J'avais également besoin que certaines machines locales soient « vues » du réseau global (bien que filtrées au travers du pare−feu de manière permanente). Ceci éliminait le masquage IP (voir le IP Masquerade HOWTO) qui, autrement, serait probablement une solution plus aisée. Ce n'est pas si simple qu'il y paraît.

2.2. La solution

Pour arriver à notre but d'isoler un réseau local du réseau global (sur DSL) en utilisant notre linuxette, nous utiliserons deux cartes ethernet (NIC). Une de ces cartes est connectée au réseau local, l'autre au réseau global. La seule machine qui peut directement communiquer avec l'extérieur est la machine Linux. Toutes les autres machines de notre réseau local doivent passer par cette dernière (pare−feu).

La configuration logicielle consiste en résoudre deux problèmes :

- router les paquets entre les réseaux local et global (pontage) ;
- filtrer les paquets pour en stopper certains lorsqu'ils transitent par le pare−feu.

Le Bridging mini−Howto) donne des instructions détaillées en ce qui concerne le premier problème de routage des paquets entre les deux parties du réseau (local et global). Ceci fonctionne en positionnant les deux cartes ethernet sur le mode « promiscuous » de façon à ce qu'elles détectent les paquets sur chacune des interfaces et transfèrent ceux−ci quand ils appartiennent à l'autre côté. Ceci se fait de manière transparente : les autres ordinateurs sur le réseau ne voient même pas le pont, car celui−ci n'a même pas d'adresse IP. Mais cela ne résout pas totalement le problème. Je voulais que le pare−feu ait une adresse IP (du moins pour l'administrer à distance) et, plus ennuyeux, le code du pont dans le noyau interceptait et transférait les paquets AVANT qu'ils ne parviennent au code du pare−feu ce qui faisait que celui−ci n'avait aucun effet. Vous pouvez alternativement attribuer des adresses IP à vos cartes et continuer à les utiliser comme un pont. Bien que le Bridging mini−Howto ne le fait pas (en réalité, il utilise l'adresse de loopback), cela fonctionne bien. Cela résout un problème. En ce qui concerne le pare−feu, on se tourne vers une rustine sympa pour le noyau à http://ac2i.tzo.com/bridge_filter/ qui fait en sorte que les règles du pare−feu soient invoquées pour les paquets qui ont traversé le pont avec une nouvelle règle « bridgein ».

2.3. Vue d'ensemble de l'installation

Ce mini−HOWTO a pour but de vous aider à prendre en mains la situation où vous avez une machine Linux configurée comme une passerelle/pare−feu. Le système a deux cartes réseau. L'une des cartes est connectée au monde extérieur (dans notre cas, un modem DSL) et rien d'autre. L'autre carte est connectée à notre réseau local.

Notez que je n'ai eu d'expérience de ceci que sur mon i386 (ABIT BP6 MOBO, w/2 céléron) avec une RedHat

6.0, un noyau 2.2.13, un modem DSL connecté à un routeur et deux cartes Net Gear FA310TX. Votre cas peut être différent.

Notez également que la démarche proposée laissera votre réseau ouvert à des attaques éventuelles au démarrage (avant que le pare−feu ne soit activé). Si vous êtes très paranoïaque, vous devrez prendre des mesures pour éviter ceci.

2.4. Références

J'ai trouvé pas mal d'informations sur internet que j'ai utilisée pour faire fonctionner les choses. Une partie de cette information était utile mais inappropriée.

Le Bridging mini−HOWTO a joué un rôle décisif pour la mise en Suvre. Malheureusement sa seule utilisation ne permet pas de mettre en place un pare−feu.

Le Linux Bridge+Firewall mini−HOWTO) me paraissait, au premier abord, être pile ce dont j'avais besoin. Néanmoins, il s'avère que je pense qu'il est inapproprié. J'ai obtenu un fonctionnement relatif mais, en fin de compte, je me suis aperçu qu'il n'était pas nécessaire de scinder notre sous−réseau en deux comme il le préconise et je n'utiliserai pas cette méthode. Si vous tombez sur ce document, considérez−le avec des réserves.

La rustine Bridge Filter est la clé pour obtenir un bon fonctionnement de l'ensemble. Assez bizarrement, l'information sur la page web vous redirige vers le Bridge+Firewall mini−HOWTO. Vous n'avez pas besoin de mettre en Suvre les indications du Bridge+Firewall mini−HOWTO pour obtenir que les choses fonctionnent. Vous aurez besoin de cette rustine.

Le IPCHAINS HOWTO est essentiel pour la mise en Suvre du pare−feu en lui−même. Je ne traiterai pas de l'installation du pare−feu dans ce document ; seules les choses qui diffèrent en raison de la mise en Suvre du pontage seront abordées.

3. Marche à suivre

La démarche générale est la suivante :

- installer le matériel (et vérifier son bon fonctionnement) ;
- patcher et configurer le noyau ;
- configurer le réseau (ifconfig, route, bridging) ;
- configurer le pare−feu.

3.1. Caractéristiques de l'installation prise comme exemple

Tout le long de la procédure, je considérerai une installation avec deux cartes ethernet (NIC), un lien extérieur DSL (où le modem DSL est connecté à une des deux cartes réseau) et un réseau local connecté à l'autre carte réseau. Arbitrairement, je désignerai la carte connectée au modem DSL par « eth1 » et la carte sur le réseau local par « eth0 ». Le nommage des périphériques par le noyau dépend des connecteurs sur lesquels ils sont enfichés.

Je supposerai que l'on vous a assigné un sous−réseau d'adresses IP à 192.168.2.128−191, c'est à dire avec un masque de réseau de 255.255.255.192, et que le routeur fourni par le fournisseur DSL est à l'adresse

192.168.2.129. Ces valeurs sont des exemples arbitraires pour illustrer l'installation. J'utiliserai l'adresse 192.168.2.130 pour le pare−feu (les deux cartes), bien, qu'en fait, vous pouvez utiliser des adresses différentes pour chacune des deux cartes si vous le souhaitez.

3.2. Installation du matériel

Vous aurez besoin de deux cartes ethernet pour faire fonctionner les choses. Le plus grand problème que j'ai eu était que j'avais choisi un connecteur au hasard sur ma carte mère pour le seconde carte réseau et qu'il s'est avéré que ce slot (PCI) partageait une interruption avec celui de la première carte réseau. Je ne croyais pas que c'était un problème (en fait il y a peu d'informations sur ceci et je supposais que ça fonctionnerait correctement). Cela entraînait que les deux cartes réseau se désactivaient silencieusement (sans indiquer d'erreur) et arrêtaient d'émettre et de recevoir des paquets. Naturellement, quand vous procédez à toutes sortes de changements de configuration, c'est bien la dernière des choses dont vous avez besoin. Je ne sais pas si c'est un problème avec toutes les cartes réseau PCI ou seulement avec les nôtres mais j'aurais tendance à alerter sur les interruptions partagées. Le driver tulip, que nous utilisons, note l'IRQ de chaque carte dans le syslog au démarrage. Il y a une quantité d'informations ailleurs (reportez−vous au Ethernet−HOWTO à la section Utiliser plus d'une carte réseau par machine) à propos de la reconnaissance de deux cartes ethernet par le noyau en utilisant des options de démarrage ; néanmoins, je n'ai pas besoin de ceci (mon noyau reconnaît les deux cartes sans argument).

Ensuite, vous devez connecter la deuxième carte réseau au modem DSL (ou quoi que ce soit d'autre qui vous relie au monde extérieur) et vous assurer que cela fonctionne. Vous devriez être en mesure « d'ifconfigurer » la seconde carte ethernet sur une adresse IP adhoc et pinguer le routeur de l'autre côté du lien. Ceci permet de vérifier que vous pouvez envoyer et recevoir des paquets au travers du lien DSL. Par exemple, pour le réseau pris en illustration vous faites :

```
ifconfig eth1 192.168.2.130 netmask 255.255.255.192 broadcast 192.168.2.191
```

pour configurer la carte. Et puis

```
ifconfig eth0 down # juste pour s'assurer que cela n'interfère pas avec autre chose
ping 192.168.2.129
```

pour tester que vous pouvez bien atteindre le routeur. Pour bien faire, vous pourriez aussi vérifier que vous pouvez atteindre les machines sur votre réseau local par l'autre carte :

```
ifconfig eth1 down # juste pour s'assurer que cela n'interfère pas avec autre chose
ifconfig eth0 up
ping 192.168.2.x # où x est l'adresse d'un machine sur votre réseau local
```

À ce point, vous vous êtes assuré que l'ensemble du matériel fonctionne.

3.3. Configuration du pont

En fonction de la version de votre noyau, vous aurez besoin soit de l'ancien outil de configuration de pont (BRCFG) pour les noyaux antérieurs à la version 2.2.14, soit du nouvel outil de configuration (bridgex) pour les noyaux ultérieurs ; ces utilitaires vous permettent de contrôler la fonction de pontage à l'intérieur du noyau quand CONFIG_BRIDGE est activé. *BRCFG* est distribué sous forme de source avec des exécutables pré−compilés. Je ne sais pas sous quel noyau les exécutables ont été compilés mais j'ai obtenu des résultats différents après une recompilation avec les fichiers include de mon noyau (2.2.13). Malheureusement, pour ce faire, j'ai dû les patcher légèrement. Voici les rustines :

```
diff −C 3 −r /tmp/BRCFG/brcfg.c ./brcfg.c
*** /tmp/BRCFG/brcfg.c  Wed Feb 21 19:11:59 1996
--- ./brcfg.c   Wed Dec  8 12:52:23 1999
***************
*** 1,6 ****

! #include <sys/types.h>
! #include <sys/socket.h>
  #include <skbuff.h>

  #include "br.h"
--- 1,6 ----

! #include <types.h>
! #include <socket.h>
  #include <skbuff.h>

  #include "br.h"
```

Appliquez la rustine, recompilez *brcfg* et installez−le dans un endroit convenable (j'ai choisi */usr/bin*).

Pour les noyaux ultérieurs à la version 2.2.13, vous devez définitivement utiliser l'utilitaire bridgex. Je ne sais pas s'il fonctionne ou non avec des noyaux plus anciens. Remarquez que l'URL pour cet utilitaire peut être trouvée dans l'aide de la configuration du noyau */usr/src/linux/Documentation/Configure.help* ainsi, si l'URL mentionnée ici est erronée, jetez un coup d'Sil dans le fichier d'aide (c'est l'aide pour l'item *CONFIG_BRIDGE*. L'archive bridgex contient un exécutable déjà compilé mais vous devrez probablement le reconstruire en utilisant le Makefile également présent. Remarquez que l'utilitaire bridgex prend des arguments légèrement différents de ceux que prend le paquet BRCFG (qui seront détaillés plus tard quand je traiterai de la configuration du pont).

3.4. Configuration du noyau

Vous devrez patcher et configurer votre noyau pour le pontage et le pont filtrant (de la même manière que pour le pare−feu, le réseau, etc. si vous n'avez pas déjà ces fonctionnalités). Les items de configuration suivants seront nécessaires (au minimum) :

```
  CONFIG_EXPERIMENTAL=y
  CONFIG_BRIDGE=y
```

```
CONFIG_FIREWALL=y
CONFIG_IP_FIREWALL=y
```

Vous devriez mettre la main sur la rustine « Bridge Filter » et l'appliquer à votre noyau. Recompilez, installez votre noyau puis redémarrez.

3.5. Assemblage du tout

À ce niveau, vous devriez avoir vos deux cartes réseau en état de fonctionner, un noyau nouvellement reconfiguré et *brcfg* installé. Maintenant, vous devez construire un script de démarrage pour assembler le tout. J'ai fait cela en utilisant les scripts « façon RedHat » (*/etc/rc.d*). J'ai déclaré les adresses et masques réseau spécifiques dans */etc/sysconfig/network* :

```
GATEWAY=192.168.2.129          # adresse du routeur DSL
GATEWAYDEV=eth1                # carte réseau à laquelle le routeur est relié
ETH0_ADDR=192.168.2.130        # adresse IP de la carte sur le réseau local
ETH0_MASK=255.255.255.192      # masque du réseau local
ETH0_BROAD=192.168.2.191       # adresse de diffusion du réseau local
ETH1_ADDR=192.168.2.130        # adresse IP de la carte côté DSL ; peut être
                               # différente de ETH0_ADDR si vous voulez
ETH1_MASK=$ETH0_MASK           # masque réseau côté DSL, devrait être le même
                               # que sur eth0
ETH1_BROAD=$ETH1_BROAD         # idem pour l'adresse de diffusion
```

Ensuite, j'ai créé un script dans */etc/rc.d/init.d/bridge* pour installer le pont. J'inclus deux scripts ici. Le premier est utilisé par le vieil outil BRCF l'autre pour le plus récent bridgex. D'abord, celui pour BRCFG :

```
#!/bin/sh
#
# bridge      Ce script installe le pontage pour DSL avec BRCFG
#
# description: utilise brcfg pour activer le pontage et configure les cartes
# ethernet
# nom du processus: bridge
# config:

# localisation de la bibliothèque de fonctions
. /etc/rc.d/init.d/functions

# localisation de la configuration du réseau
. /etc/sysconfig/network

# voyons comment nous sommes appelé
case "$1" in
  start)
        echo −n "configuration du pont: "
        ifconfig eth0 $ETH0_ADDR netmask $ETH0_MASK broadcast $ETH0_BROAD
        ifconfig eth1 $ETH1_ADDR netmask $ETH1_MASK broadcast $ETH1_BROAD
        route add $GATEWAY dev $GATEWAYDEV
        route add default gw $GATEWAY dev $GATEWAYDEV
```

```
        ifconfig eth0 promisc
        ifconfig eth1 promisc
        brcfg -enable
        echo
        ;;
  stop)
        # arrêt des démons
        brcfg -disable
        ifconfig eth0 down
        ifconfig eth1 down
        ;;
  restart)
        $0 stop
        $0 start
        ;;
  status)
        ifconfig eth0
        ifconfig eth1
        brcfg
        ;;
  *)
        echo "utilisation: bridge {start|stop|restart|status}"
        exit 1
esac

exit 0
```

Le script qui suit est à utiliser avec l'utilitaire de configuration de pont bridgex. Notez que bridgex est bien plus configurable que le moins jeune BRCFG et que vous devriez jeter un coup d'Sil à la page de manuel incluse dans l'archive de bridgex et adapter ce script :

```
#!/bin/sh
#
# bridge      Ce script installe le pontage pour DSL avec bridgex
#
# description: utilise bridgex pour démarrer le pontage et configurer les
# cartes ethernet
# nom du processus: bridge
# configuration:

# localisation de la bibliothèque de fonctions
. /etc/rc.d/init.d/functions

# localisation de la configuration du réseau
. /etc/sysconfig/network

# voyons comment nous sommes appelé
case "$1" in
  start)
        echo -n "configuration du pont: "
        ifconfig eth0 $ETH0_ADDR netmask $ETH0_MASK broadcast $ETH0_BROAD
        ifconfig eth1 $ETH1_ADDR netmask $ETH1_MASK broadcast $ETH1_BROAD
        route add default gw $GATEWAY dev $GATEWAYDEV
        ifconfig eth0 promisc
        ifconfig eth1 promisc
        brcfg start
        brcfg device eth0 enable
        brcfg device eth1 enable
```

```
        echo
        ;;
  stop)
        # arrêt des démons
        brcfg stop
        ifconfig eth0 down
        ifconfig eth1 down
        ;;
  restart)
        $0 stop
        $0 start
        ;;
  status)
        ifconfig eth0
        ifconfig eth1
        brcfg
        ;;
  *)
        echo "utilisation: bridge {start|stop|restart|status}"
        exit 1
esac

exit 0
```

Le script est exécuté lors du démarrage. Il attribue les adresses à chacune des cartes réseau, ajoute une route par défaut qui pointe vers le routeur DSL, ajoute une route spécifique vers le routeur DSL, positionne chaque carte en mode « promiscuous » et puis active le pontage. J'ai créé des liens vers ce script dans les répertoires suivants dans */etc/rc.d* :

```
/etc/rc.d/rc0.d/K90bridge
/etc/rc.d/rc1.d/K90bridge
/etc/rc.d/rc2.d/S11bridge
/etc/rc.d/rc3.d/S11bridge
/etc/rc.d/rc4.d/S11bridge
/etc/rc.d/rc5.d/S11bridge
/etc/rc.d/rc6.d/K90bridge
```

Ceci fait en sorte qu'il s'exécute après le script de démarrage du réseau. Vous devez désactiver les autres configurations de eth0 (ou eth1) telles que celles qui sont faites dans le script */etc/rc.d/init.d/network* (en supprimant les fichiers *ifcfg−eth?* de */etc/sysconfig/network−scripts/* dans la RedHat).

Pour tester les choses, je suggère de redémarrer en mode mono−utilisateur (en spécifiant « single » comme argument du noyau, par exemple, dans lilo « lilo: linux single ») et en exécutant les scripts de démarrage de */etc/rc.d/rc3.d* un par un jusqu'à ce que vous arriviez au démarrage du pont. Démarrez le pont et alors vérifiez si vous pouvez atteindre des machines (vous utiliserez certainement « *ping −n* » pour ceci, afin de maintenir le serveur de noms hors jeu) :

- pinguez le routeur DSL ;
- pinguez une machine locale ;

- pinguez une machine sur le réseau global.

Si vous pouvez pinguer toutes ces localisations, il y a de fortes chances que tout se passe bien. Remarquez que le pont met un petit moment à démarrer. Vous pouvez contrôler son état en utilisant la commande *brcfg* sans option.

3.6. Installation du pare−feu

Vous aurez besoin d'installer un pare−feu (en supposant que vous en voulez un) pour vous protéger des accès non autorisés. La rustine « Bridge Filter » vous permet d'utiliser une nouvelle règle intégrée « bridgein » avec ipchains. Cette règle est utilisée à chaque fois qu'un paquet doit être redirigé de eth0 vers eth1 ou inversement. La règle bridgein n'est pas utilisée quand le paquet est destiné au pare−feu lui même ; vous devez utiliser la règle input pour cela. Je ne tenterai pas de rentrer en détail dans l'installation du pare−feu. Référez−vous au IPCHAINS HOWTO pour ceci.

3.7. Installation d'une machine locale

Pour chacune des machines de votre réseau local, vous devez simplement lui indiquer une adresse IP, un masque de réseau et utiliser le routeur DSL comme une passerelle (route par défaut) le pont filtrant transférera les paquets vers et depuis le routeur DSL.

4. Bizarreries et problèmes

4.1. Message étrange à l'utilisation d'ipchains −X

La rustine qui ajoute la règle intégrée bridgein à ipchains fait que la commande de suppression de toutes les chaînes, *ipchains −X*, produit l'erreur suivante :

```
ipchains: Device or resource busy
```

d'autant que je sache, ceci n'est pas grave. Je soupçonne que ipchains ne comprend pas que la nouvelle règle d'entrée de pont est prédéfinie.

4.2. Interruptions partagées

Comme je l'ai mentionné dans la section Section 3.2, au moins pour les cartes PCI, vous ne devez pas partager d'interruptions entre deux cartes (et probablement aussi avec aucun autre périphérique).

5. Adaptation française

5.1. Traduction

La traduction française de ce document a été réalisée par Geneviève Gracian `<ggracian@free.fr>`.

5.2. Relecture

La relecture de ce document a été réalisée par XavierVenient <xvenient@free.fr>.