Netfilter / IPtables

Préambule.

On peut parler à l'infini des avantages comparés de Linux face aux systèmes Microsoft et réciproquement, il semble même possible de réussir à déclencher quelques guerres de religion à ce sujet, il y a tout de même un domaine où la discussion est impossible, sauf peut être avec une mauvaise foi extrême, c'est celui de la gestion des réseaux. Même si Windows 2000 est en très net progrès par rapport à Windows NT4 (nous parlons ici des versions "server"), Les noyaux Linux 2.4.x avec IPTables et IPRoutes2 disposent, à mon sens, d'une très confortable avance.

Ces versions 2.4 du noyau Linux sont particulièrement riches dans ce domaine et il devient sans doute possible de réaliser avec ce système des passerelles et des firewalls aussi performants sinon plus, que certains matériels spécialisés.

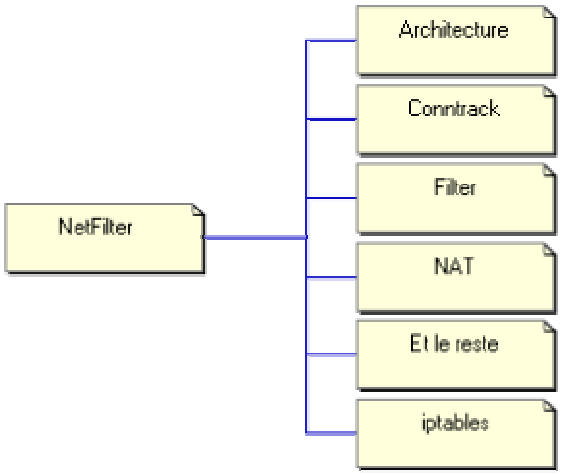
Iproutes2 constitue une nouvelle approche de la gestion des routes inter réseaux. Iproutes2 mériterait à lui seul un chapitre complet. Nous ne ferons ici que l'évoquer.

Netfilter permet de faire beaucoup plus de choses en matière de filtrage de paquets et de translation d'adresses que ses prédécesseurs, ce qui fait de Linux 2.4 un outil de choix pour la réalisation de passerelles entre réseaux privés et l'Internet.

Ce chapitre a pour but de présenter succinctement l'architecture et les principales fonctionnalités de Netfilter, dans le cadre d'un réseau domestique connecté au Net par une ligne "haut débit" type câble ou ADSL . Si vous voulez absolument tout savoir sur Netfilter, il vous faudra chercher également ailleurs. La première source à consulter (de toutes manières, je n'ai rien inventé, je me suis contenté d'essayer de rendre plus "digestes" les documents initiaux), c'est:

- Rusty's Remarkably Unreliable Guides
  Ce sont les documentations "officielles". Vous y trouverez même des documents traduits en français.

Plan du chapitre.



Architecture

Avant propos.

Avertissement important à ceux qui maîtrisent IPchains.

NetFilter avec IPtables n'a pas du tout la même architecture, le fonctionnement est différent, même si la syntaxe d'IPtables peut paraître proche de celle d'IPchains.

En particulier, les chaînes INPUT et OUTPUT ne contrôlent pas tout ce qui entre et sort de la passerelle, routage compris, mais uniquement ce qui entre en direction de la passerelle elle même et sort de la passerelle elle même. Entendez par là que tout le trafic entre le réseau privé masqué par le NAT et l'Internet ne sera aucunement influencé par ces deux chaînes. Nous verrons en détail plus loin pourquoi et comment.

Finalement, le meilleur conseil à donner aux utilisateurs d'IPchains, c'est d'oublier purement et simplement tout ce qu'ils savent sur le sujet...

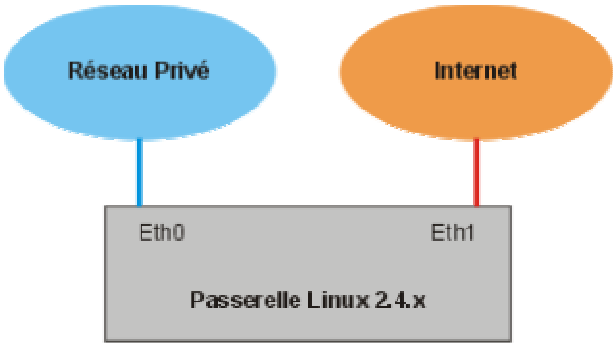
Avertissement aux utilisateurs exclusifs des interfaces graphiques.

Avec les versions 2.2.x, nous avions l'excellent outil "gfcc" qui permettait de faire à peu près tout ce qu'il était possible de faire avec IPchains. Malheureusement, avec IPtables, je n'ai pas encore trouvé d'outil équivalent à l'heure où j'écris ces lignes. Essayez de voir avec knetfilter, mais ce n'est pas du tout la même chose.

Position du problème.

Comme d'habitude sur ce site, l'exposé s'appuie sur une passerelle Linux mettant en relation un réseau privé avec le Net, au moyen d'une liaison "haut débit" et "permanente" de type câble ou ADSL.

Topologie de la machine Linux.



La machine Linux dispose de deux interfaces Ethernet. Ici:

- Eth0 sur le réseau local (privé)
- Eth1 sur l'Internet via le modem câble du fournisseur d'accès. Cette interface, le plus souvent, sert de support à PPPoE.

Il faut bien comprendre qu'à priori, il peut entrer et sortir des données de chaque interface.

- Il peut entrer et sortir des données par Eth0, parce qu'un client du réseau local établit une connexion avec un service installé sur la passerelle, Samba par exemple, pour le partage de fichiers avec Windows.
- Il peut entrer et sortir des données par Eth1 (ou la connexion ppp qui y est associée) parce qu'un client situé sur le Net établit une connexion avec un service installé sur la passerelle, une session ssh ou telnet par exemple (ssh, c'est mieux...). Il peut se faire aussi que cette connexion s'établisse à votre insu, parce qu'un pirate est en train de prendre possession de votre machine.
- Il peut se faire également qu'une connexion s'établisse entre un poste du réseau privé et un serveur situé sur le Net. Dans ce cas, les paquets entreront par une interface et sortiront par l'autre.
- En toute rigueur, il est également possible qu'une connexion s'établisse entre un client situé sur le Net et un serveur situé sur votre réseau privé (si si, c'est possible aussi, bien que dans le cadre d'un réseau domestique, ce ne soit pas nécessaire, ni même souhaitable). Là aussi, les paquets qui entrent par une interface sortiront par l'autre.

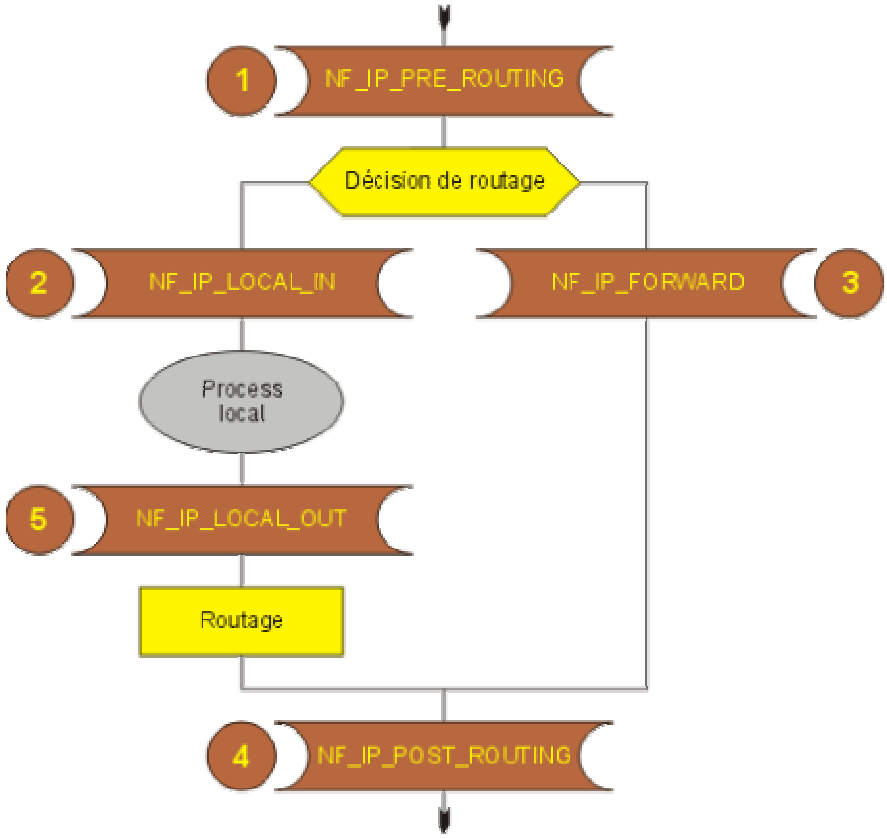
Quel que soit le cas de figure vu plus haut, dans ce qui suit, nous ne ferons pas de ségrégation sur l'interface par laquelle les paquets entrent ni l'interface par laquelle les paquets sortent. Cette ségrégation interviendra éventuellement dans les règles que nous écrirons avec IPTables.

Netfilter dans la pile IP.

En tout état de cause, dans l'explication qui suit, quelles que soient l'origine et la destination des paquets, ils vont entrer dans la pile de protocoles IP par le même point et en sortir par le même autre point.

Netfilter se présente comme une série de 5 "hooks" (points d'accrochage), sur les quels des modules de traitement des paquets vont se greffer. Ces points sont:

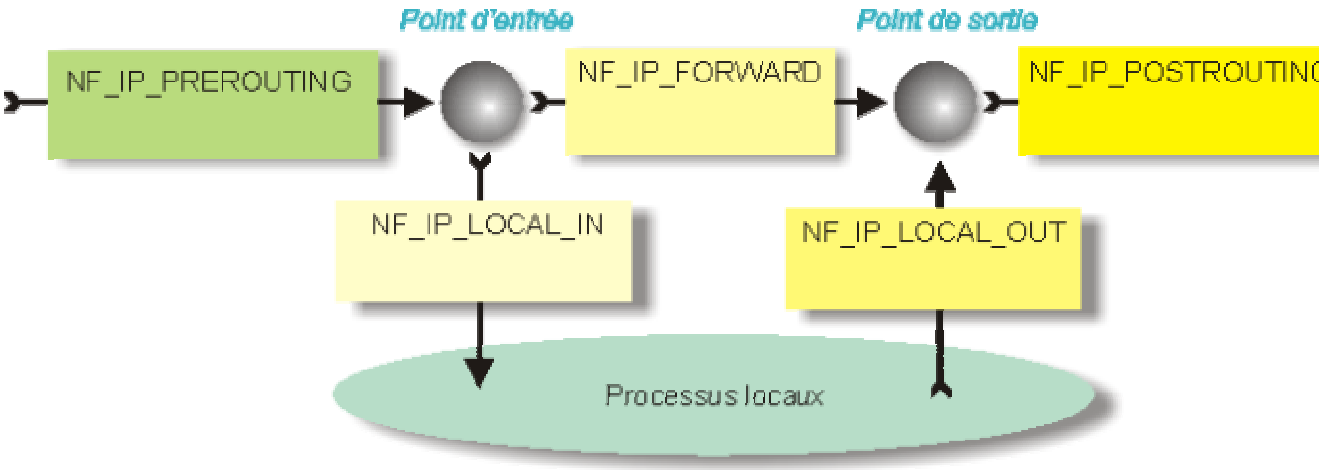
- NF_IP_PRE_ROUTING
- NF_IP_LOCAL_IN
- NF_IP_FORWARD
- NF_IP_POSTROUTING
- NF_IP_LOCAL_OUT



La branche gauche représente le trajet des paquets qui entrent et qui sortent vers et depuis un processus local (SMB, FTP, HTTP etc.)

La branche de droite représente le trajet des paquets qui traversent notre passerelle dans sa fonction de routeur.

Que l'on peut représenter autrement ainsi :



Attention toutefois, ces diagrammes sont faux, dans la mesure où ils sont largement simplifiés. Leur but est uniquement de montrer où Netfilter vient interagir avec la pile IP, en aucun cas il ne représente l'architecture complète de la pile IP.

Ce que sait faire Netfilter.

A travers ces cinq points d'insertion, Netfilter va être capable :

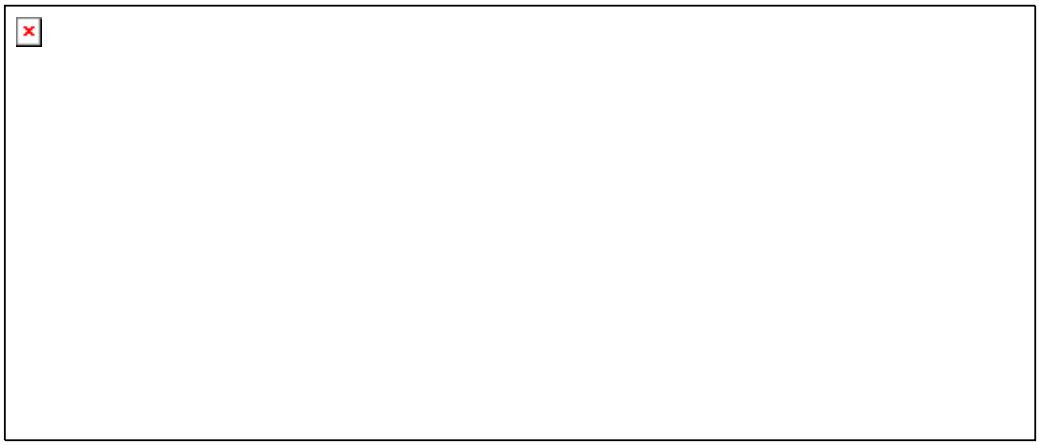
- D'effectuer des filtrages de paquets, principalement pour assurer des fonctions de Firewall. On pourra par exemple interdire à tous les paquets venant de l'Internet et s'adressant au port 80 (HTTP) de passer. Notre serveur APACHE est un serveur Intranet et ne doit pas être accessible depuis l'extérieur.
- D'effectuer des opérations de NAT (Network Address Translation) Ces fonctions sont particulièrement utiles lorsque l'on veut faire communiquer tout ou partie d'un réseau privé, monté avec des adresses IP privées (192.168.x.x par exemple) avec l'Internet.
- D'effectuer des opérations de marquage des paquets, pour leur appliquer un traitement spécial. Ces fonctionnalités sont particulièrement intéressantes sur une passerelle de réseau d'entreprise, un peu moins pour notre ces de réseau domestique.

Comment Netfilter sait le faire.

Netfilter dispose d'une commande à tout faire : IPtables. Cette commande va permettre, entre autres, d'écrire des chaînes de règles dans des tables. Il y a dans Netfilter trois tables qui correspondent aux trois principales fonctions vues plus haut:

Les tables et leurs chaînes.

Il existe trois tables qui vont servir à contenir des règles de "filtrage":



La table "Filter".

Cette table va contenir toutes les règles qui permettront de filtrer les paquets. Cette table contient trois chaînes:

- **La chaîne INPUT.**
  Cette chaîne décidera du sort des paquets entrant **localement** sur l'hôte.
- **La chaîne OUTPUT.**
  Ici, ce ne sont que les paquets émis par **l'hôte local** qui seront filtrés

- **La chaîne FORWARD.**
  Enfin, les paquets qui traversent l'hôte, suivant les routes implantées, seront filtrés ici.

La table NAT.

Cette table permet d'effectuer toutes les translations d'adresses nécessaires.

- **La chaîne PREROUTING.**
  Permet de faire de la translation d'adresse de destination. Cette méthode est intéressante si l'on veut faire croire au monde extérieur, par exemple, qu'il y a un serveur WEB sur le port 80 de la passerelle, alors que celui-ci est hébergé par un hôte du réseau privé, sur le port 8080.
- **La chaîne POSTROUTING.**
  Elle permet de faire de la translation d'adresse de la source, comme du masquage d'adresse, la méthode classique pour connecter un réseau privé comme client de l'Internet, avec une seule adresse IP "officielle".
- **La chaîne OUTPUT.**
  Celle-ci va permettre de modifier la destination de paquets générés localement (par la passerelle elle-même).

La table MANGLE.

Cette table permet le marquage des paquets entrants (PREROUTING) et générés localement (OUTPUT). Le marquage de paquets va permettre un traitement spécifique des paquets marqués dans les tables de routage avec IPROUTE 2. Ceci nous mènerait trop loin. Si vous êtes intéressé par cette question, voyez à ce sujet le document: "Linux 2.4 Advanced Routing HOWTO".

Depuis la version 2.4.18 du noyau, d'autres tables ont été rajoutées sur tous les "hooks". Nous avons ainsi à notre disposition les tables supplémentaires INPUT, POSTROUTING et FORWARD

Les chaînes.

Les chaînes sont des ensembles de règles que nous allons écrire dans chaque table. Ces chaînes vont permettre d'identifier des paquets qui correspondent à certains critères.

Les cibles.

Les cibles enfin sont des sortes d'aiguillage qui dirigeront les paquets satisfaisant aux critères . Les cibles préconstruites sont :

- ACCEPT
  Les paquets qui satisfont aux critères sont acceptés, ils continuent leur chemin dans la pile,
- DROP
  Les paquets qui satisfont aux critères sont rejetés, on les oublie, on n'envoie même pas de message ICMP . Un trou noir, quoi.
- LOG
  C'est une cible particulière qui permet de tracer au moyen de syslog les paquets qui satisfont aux critères.

Suivant les contextes, d'autres cibles deviennent accessibles, comme REJECT (similaire à DROP, mais avec envoi d'un message d'erreur ICMP à la source du paquet rejeté), RETURN, REDIRECT, SNAT, DNAT, MASQUERADE...

En français, nous pourrons faire des choses de ce genre :

- Par défaut, tous les paquets qui entrent sont rejetés (DROP)
- Les paquets qui entrent par le port 80 sur l'interface eth0 sont acceptés
- Les paquets qui sortent par le port 80 sur l'interface eth0 sont acceptés
- etc.

Nous verrons cela plus en détail dans les divers exemples.

La commande IPtables.

IPtables est en quelques sortes l'interface utilisateur de Netfilter. Dans sa partie "visible", ça ressemble à IPchains, mais ici, ce n'est qu'une interface de commande de Netfilter. La syntaxe est plus complète et plus rigoureuse.

Conntrack

Le suivi de connexion.

Le suivi de connexion est un concept essentiel dans Netfilter. C'est une sorte d'intelligence artificielle qui permet d'établir des liens de cause à effet entre les paquets qui passent dans la pile. Il faut, à un moment donné, dire quelques mots à propos de ce suivi de connexion. Comme c'est une notion qui va intervenir aussi bien dans le filtrage que dans la traduction d'adresses, autant en parler tout de suite.

Expérience préliminaire.

Le principe du suivi de connexion permet de réaliser un "firewall statefull", c'est à dire qu'il va réagir intelligemment sur une connexion donnée, suivant son état. Comme ce n'est pas très simple d'expliquer ça, nous allons observer un exemple sur le terrain. Nous allons établir une connexion TCP à partir du protocole HTTP :

No.	Time	Source	Destination	Protocol	Info
10	10.181832	192.168.0.10	212.27.35.1	TCP	4252 > http [**SYN**]
11	10.204707	212.27.35.1	192.168.0.10	TCP	http > 4252 [**SYN, ACK**]
12	10.204848	192.168.0.10	212.27.35.1	TCP	4252 > http [**ACK**]
13	10.205333	192.168.0.10	212.27.35.1	HTTP	GET / HTTP/1.1

L'établissement d'une connexion TCP suit un protocole strict :

- Une requête de synchronisation [SYN] de la part de l'initiateur du dialogue (le client),
- une réponse d'accusé réception de la synchronisation [SYN,ACK] de la part du serveur,
- un accusé réception du client [ACK]

Observons également les sockets.

- Trame n°10:
  Le client [192.168.0.10] s'adresse au serveur [212.27.35.1] sur le port http (80). Il attend une réponse sur le port 4252. C'est un numéro pris plus ou moins au hasard. A priori, personne ne peut le deviner à l'avance.
- Trame n°11:
  Le serveur, bien élevé, répond au client sur le port demandé (4252)

La même chose, mais plus détaillée, ce qui donne l'occasion de voir l'ensemble des "flags" exploitables au niveau TCP :

```
Frame 10 (62 bytes on wire, 62 bytes captured)
...
Transmission Control Protocol
   Source port: 4252 (4252)
   Destination port: http (80)
   Sequence number: 3290220049
   Header length: 28 bytes
   Flags: 0x0002 (SYN)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...0 .... = Acknowledgment: Not set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..1. = Syn: Set
      .... ...0 = Fin: Not set
...
Frame 11 (62 bytes on wire, 62 bytes captured)
...
Transmission Control Protocol
   Source port: http (80)
   Destination port: 4252 (4252)
   Sequence number: 1602605975
   Acknowledgement number: 3290220050
   Header length: 28 bytes
   Flags: 0x0012 (SYN, ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..1. = Syn: Set
      .... ...0 = Fin: Not set
...
Frame 12 (54 bytes on wire, 54 bytes captured)
...
Transmission Control Protocol
   Source port: 4252 (4252)
   Destination port: http (80)
   Sequence number: 3290220050
   Acknowledgement number: 1602605976
   Header length: 20 bytes
   Flags: 0x0010 (ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
```

```
      ...1 .... = Acknowledgment: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
   Window size: 16944
   Checksum: 0xe79b (correct)
```

De ces premières observations, nous pouvons déduire quelques choses intéressantes. En considérant des échanges TCP entre deux sockets toujours les mêmes :

- Lorsqu'un paquet TCP contient le flag SYN, c'est que c'est une nouvelle connexion qui commence.
- lorsqu'un paquet TCP contient les flags SYN et ACK, c'est que la connexion est acceptée, elle est donc établie,
- lorsqu'un paquet ne contient que le flag ACK, c'est que la connexion se continue.

Mais voyons un peu plus loin...

No.	Time	Source	Destination	Protocol	Info
29	10.427697	192.168.0.10	212.27.35.1	TCP	4252 > http [ACK]
30	10.731147	**192.168.0.10**	**212.27.35.1**	TCP	**4253** > http [**SYN**]
31	10.752981	212.27.35.1	192.168.0.10	TCP	http > **4253** [**SYN, ACK**]
32	10.753165	192.168.0.10	212.27.35.1	TCP	**4253** > http [**ACK**]
33	10.753707	192.168.0.10	212.27.35.1	HTTP	GET /images/titre.gif HTTP/1.1
34	10.780941	192.168.0.10	212.27.35.1	TCP	**4252** > http [**FIN, ACK**]

Ce que nous observons ici, c'est l'établissement d'une nouvelle connexion TCP entre les mêmes protagonistes. Bien entendu pour éviter les mélanges, le port du client n'est plus le même. C'est cette fois-ci 4253.

Autrement dit, le même client (192.168.0.10) ouvre une nouvelle connexion TCP sur le même serveur (212.27.35.1), toujours sur le port 80, mais attend les réponses sur un nouveau port, alors que la connexion précédente existe toujours.

Le client met fin à la précédente (trame 34) avec le flag [FIN] une fois seulement que la seconde connexion est établie.

Dans ces conditions, il est pertinent de penser que cette nouvelle connexion est en relation directe avec la première. Nous dirons que c'est une connexion en relation avec la première.

Premières déductions.

Si l'on met en place un système capable de mémoriser ce qu'il se passe sur la couche TCP, alors il va devenir possible de savoir si une connexion est dans l'un de ces états :

- NEW
  nouvelle connexion (elle contient le flag SYN),
- ESTABLISHED
  connexion déjà établie, elle ne devrait pas contenir de SYN ni de FIN,
- RELATED
  la connexion présente une relation directe avec une connexion déjà établie,
- INVALID

- la connexion n'est pas conforme, contient un jeu de flags anormal, n'est pas classable dans l'une des trois catégories précédentes.

Ceci est intéressant, parce que l'on pourra interdire à priori à toute connexion NEW d'entrer dans notre installation, il n'y a aucune raison qu'il en rentre si nous n'avons pas de serveur. Eventuellement, nous pourrons par exemple créer des exceptions pour les port ssh, si nous souhaitons accéder à notre machine depuis le Net.

En revanche, toute connexion ESTABLISHED ou RELATED devra pouvoir entrer depuis le Net.

Dans l'autre sens, du LAN vers le Net, les paquets NEW doivent pouvoir passer, de même, bien entendu que les ESTABLISHED et les RELATED.

Les paquets INVALID pourront éventuellement être tracés dans les logs.

Et pour l'UDP ?

Là, c'est plus délicat puisqu'il n'y a justement pas de connexion. Il sera donc impossible de définir de façon précise l'état d'un échange UDP. Ce que l'on pourra faire, c'est mettre en place un "timer" pour décider de l'état d'un paquet UDP. Nous pouvons prendre l'exemple simple d'une requête DNS depuis notre réseau privé.

- Le premier paquet UDP sort de notre réseau, sur un port connu et identifié (53) vers un serveur DNS. Nous pouvons décider de le laisser passer et nous le qualifions de "NEW". Il déclenche un timer,
- si avant expiration du timer, nous recevons un paquet UDP dudit serveur DNS, nous considèrerons que c'est un paquet "ESTABLISHED".

Dans la pratique.

Un module principal de suivi de connexion est chargé dynamiquement en cas de besoin, il s'agit du module ip_conntrack. Cependant tout n'est pas toujours si simple et ce module peut montrer ses limites sur des protocoles particulièrement complexes, comme par exemple FTP.

ip_conntrack ne pourra assurer qu'une connexion FTP de type passive. Si l'on souhaite assurer le suivi de connexion sur du FTP en mode actif, il faudra avoir recours au module spécialisé ip_conntrack_ftp. Mais celui-ci ne se chargera pas dynamiquement, vous aurez à le charger vous-même. Nous aurons aussi besoin dans ce cas de ip_nat_ftp; si notre passerelle fait du NAT.

Nous verrons sur le terrain le travail de conntrack dans divers exemples.

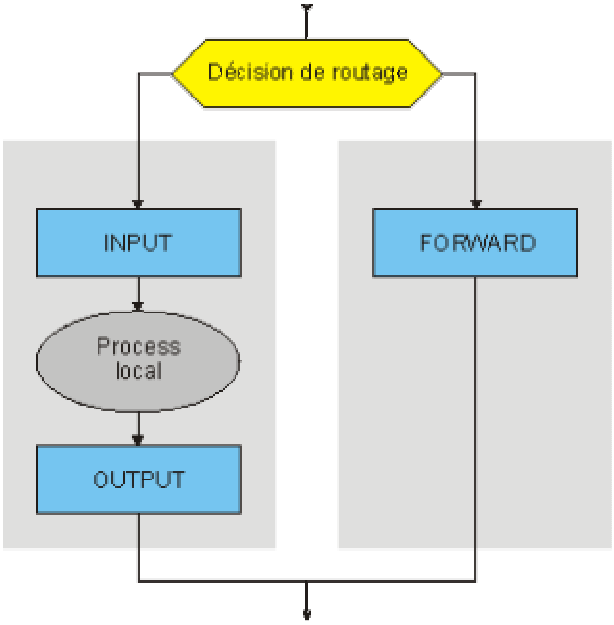
Filter

La table de filtrage.

C'est la table qui va permettre de filtrer tous les paquets qui entrent et sortent de notre machine. Il n'y a ici aucune modification de ces paquets, ils seront comparés à des

critères définis dans la table Filter. Dans notre cas, il peut se passer deux choses différentes:

- Un paquet qui entre est destiné à un processus de l'hôte (serveur HTTP, FTP...).
- Un paquet qui entre est destiné à un autre réseau, c'est alors une fonction de routage.



Un paquet entre dans notre machine. Peu importe par quelle interface il entre, il peut venir aussi bien du réseau local que de l'Internet. Il passe d'abord par la fonction de décision de routage. C'est elle qui va déterminer si le paquet est destiné à un processus local de l'hôte ou à un hôte sur un autre réseau.

- Si le paquet est destiné à l'hôte local:
  - Il traverse la chaîne INPUT
  - S'il n'est pas rejeté, il est transmis au processus impliqué. Ce processus va donc le traiter et éventuellement émettre un nouveau paquet en réponse.
  - Ce nouveau paquet traverse la chaîne OUTPUT
  - S'il n'est pas rejeté, il va vers la sortie.
- Si le paquet est destiné à un hôte d'un autre réseau:
  - Il traverse la chaîne FORWARD.
  - S'il n'est pas rejeté, il poursuit alors sa route.

Une autre façon de représenter graphiquement tout ça serait la suivante :

La chaîne INPUT sera raccrochée au "hook" NF_IP_LOCAL_IN
la chaîne OUTPUT au "hook" NF_IP_LOCAL_OUT
et la chaîne FORWARD à NF_IP_FORWARD.

Rappel d'avertissement important:

Pour ceux qui ont travaillé avec IPchains, notez que la démarche est ici différente et ça va peut-être vous poser pas mal de problèmes...

Avec IPChains...	**Avec IPtables...**
TOUS les paquets entrants passaient par les chaînes INPUT qu'ils soient destinés à un process local où au routage	SEULS les paquets destinés à un process local traversent la chaîne INPUT
TOUS les paquets sortants passaient par la chaîne OUTPUT, qu'ils soient issus d'un process local ou destinés au routage.	SEULS les paquets issus d'un process local traversent la chaîne OUTPUT
	SEULS les paquets destinés au routage traversent la chaîne FORWARD

L'illustration ci-dessus le montre bien, et ceci va conduire à d'énormes erreurs, si l'on se contente de traduire les anciennes règles IPChains en règles IPTables, sans précautions particulières.

Vous ne me croyez pas ?

Alors, avant d'aller plus loin, faites la manip suivante. Sur votre passerelle Linux 2.4.x:

Initialisation de Netfilter.

- Tapez successivement les commandes suivantes pour initialiser votre système:
- iptables -F
- iptables -X
- iptables -t nat -F

iptables -t nat -X

De cette manière, vous avez toutes vos chaînes vides, avec par défaut la règle "ACCEPT"

iptables -L

L'affichage va vous assurer que les trois chaînes INPUT, FORWARD et OUTPUT sont vides et on bien la règle par défaut ACCEPT

iptables -t nat -L

L'affichage va vous assurer que les trois chaînes PREROUTING, POSTROUTING et OUTPUT sont vides et ont bien la règle par défaut ACCEPT.

Mise en place de Masquerade

- Tapez maintenant les commandes suivantes:

  iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

  Où ppp0 représente l'interface connectée à l'Internet. Remplacez éventuellement, si votre configuration est différente. Ceci signifie en gros: Tout ce qui sort du routage (-A POSTROUTING) et qui doit passer vers l'Internet (-o ppp0) doit subir un masquage d'adresse (-j MASQUERADE)

  echo 1 > /proc/sys/net/ipv4/ip_forward

  Ceci pour être certain que votre noyau autorise le routage. Vous n'en avez pas besoin, si votre machine est configurée par défaut pour assurer le routage.

Voilà. Votre passerelle entre le réseau privé et l'Internet doit être opérationnelle.

- Vérifiez que, depuis votre passerelle Linux, vous avez bien accès au Net.
- Vérifiez que, depuis un poste de votre réseau privé, vous avez bien accès au Net.

Attention, dans cet état, vous n'avez rigoureusement aucune défense contre d'éventuelles intrusions ! Passez rapidement à la suite.

Et maintenant, la manip décisive...

Pour bien montrer que les chaînes INPUT et OUTPUT n'interviennent pas dans le routage, nous allons tout simplement leur mettre DROP comme règle par défaut. Attention, il faut que vos clients utilisent un DNS situé ailleurs que sur votre passerelle Linux, sinon, ça ne fonctionnera pas à cause du DNS.

- Tapez les commandes suivantes:
- iptables -P INPUT DROP
- iptables -P OUTUT DROP

iptables -L (pour vérifier que INPUT et OUTPUT "droppent" bien tout ce qui passe).

- Vérifiez que, depuis votre passerelle Linux, vous n'avez plus accès au Net (ni à votre réseau privé d'ailleurs).
- Vérifiez que, depuis un poste de votre réseau privé, **vous avez toujours l'accès au Net**.

Convaincu ? **INPUT et OUTPUT n'interviennent absolument pas dans le routage**. Toutes les règles que vous pourrez y mettre ne concerneront que la sécurité de la passerelle elle-même, mais pas de votre réseau privé.

NAT

La table de translation d'adresses.

Remarques importantes.

La traduction d'adresse (NAT comme Network Address Translation) est à prendre ici au sens le plus large, puisque cette table permet non seulement de faire de la translation stricte d'adresses, mais également de la translation de ports et un mélange des deux, dont le masquage d'adresse est une forme particulière.

Mais qu'est-ce que c'est exactement ?

Dans un datagramme, en plus des données, on y trouve également quelques informations concernant le protocole utilisé et des identificateurs de l'émetteur et du destinataire. Ce sont ces identificateurs qui nous intéressent:

- L'adresse IP du destinataire.
- Le port du service utilisé sur le destinataire.

Ces informations constituent une "socket", elles sont indispensables pour arriver à joindre le bon service sur le bon serveur, par exemple le service HTTP du serveur www.wanadoo.fr.

- L'adresse IP de l'émetteur.
- Le port de réponse.

Ces informations constituent une autre "socket", elles sont indispensables pour que l'émetteur d'un paquet puisse espérer recevoir une réponse.

Avec les fonctions NAT de Netfilter, Lorsqu'un paquet transite par notre passerelle, nous allons pouvoir "bricoler" ces sockets absolument comme on veut. Par exemple, nous pourrons changer l'adresse de l'émetteur ou le port de l'émetteur ou les deux. Nous pouvons aussi changer l'adresse du destinataires, ou le port du destinataire, ou les deux.

Mais à quoi ça sert ?

Ca sert à une quasi infinité de choses. Parmi les plus intéressantes, citons:

Le masquage d'adresse.

C'est une fonction fondamentale lorsque l'on souhaite connecter un réseau privé à l'Internet lorsque l'on ne dispose que d'une seule IP valide sur le Net, même si celle-ci est dynamique, ce qui est le cas qui nous intéresse le plus. Les clients sont sur le réseau privé et les serveurs sont sur le Net. C'est une forme particulière de SNAT (Source NAT)

C'est ce que sont capables de faire tous les routeurs SOHO (Small Office, Home Office) qui permettent de relier un petit réseau local à l'Internet, lorsque l'on ne dispose que d'un accès RTC, NUMERIS, Câble, ADSL... Un simple (très) vieux PC (un 486 suffit) équipé d'un Linux 2.4.x permet de le faire aussi bien sinon mieux.

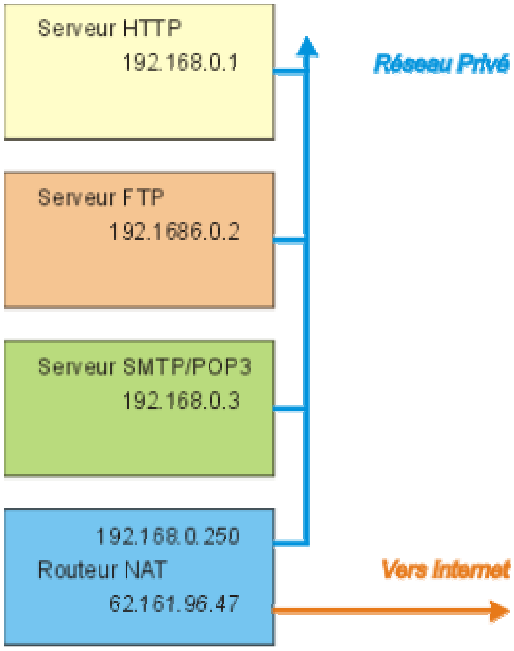
Le NAT de destination.

Ici, c'est pour résoudre les problèmes qui apparaissent dans l'autre sens. Les clients sont sur le Net et les serveurs sont sur le réseau privé.

Imaginons que nous n'ayons qu'une seule IP valide sur le Net et que nous voulions tout de même offrir des services tels que HTTP, FTP, SMTP, POP et peut-être d'autres encore. Comment faire ? La réponse triviale consiste à dire: "J'ai droit à une seule IP, donc je place tous ces serveurs sur la même machine, celle qui a la seule IP à laquelle j'ai droit."

Oui, mais la démarche est simpliste:

- Comment assurer un minimum de sécurité sur une machine ouverte de tous les côtés ?
- Comment faire pour assurer une disponibilité suffisante à chaque service dans les montées en charge ?



Cette solution ne parait finalement pas très acceptable, mais comment faire autrement ? Tout simplement avec NAT. La machine frontale sera un simple routeur NAT. Côté Internet, elle possède la seule IP valide disponible, elle va faire croire que tous les services sont dessus, mais en réalité, lorsqu'elle va recevoir un paquet dont le socket de destination est 62.161.96.47:80, elle va remplacer ça vite fait par 192.168.0.1:80 et router le paquet vers le serveur HTTP. Lorsque la réponse du serveur va lui parvenir, elle remplacera le socket de l'émetteur (192.168.0.1:80) par 62.161.96.47:80 et enverra ça sur le Net. Tout le monde n'y verra que du feu.

Bien entendu, le routeur NAT est capable de faire ça pour chacun des autres serveurs:

- Ce qui lui arrivera sur les ports 20 et 21 sera redirigé sur le serveur FTP (en réalité, le cas du FTP est bien plus difficile à résoudre que ça. Il faudra d'abord lire le chapitre sur FTP).
- Ce qui lui arrivera sur le port 25 sera redirigé sur le serveur SMTP/POP3 service SMTP
- Ce qui lui arrivera sur le port 110 sera redirigé sur le serveur SMTP/POP3 service POP3

Le cas du proxy transparent.

Bien qu'à priori, cette possibilité soit sans intérêt sur un réseau domestique, je préfère en parler parce que ce sujet peut revêtir une certaine gravité quant aux atteintes aux libertés individuelles.

Tout le monde sait ce qu'est un proxy (serveur mandataire, en français) ? C'est un serveur auquel on s'adresse pour qu'il nous fournisse des informations situées sur un autre serveur, sur les protocoles HTTP et FTP essentiellement.. Le principal avantage d'un proxy est qu'il garde en mémoire dans un cache toutes les informations qu'il est déjà allé

chercher. Si, sur un réseau privé, 10 personnes cherchent la même information, elle ne sera téléchargée qu'une fois sur le Net. L'avantage évident est l'optimisation de la bande passante sur le lien Internet, lorsque le réseau privé est un peu conséquent. L'autre avantage, c'est que l'on peut réaliser un "firewall applicatif" pour le protocole HTTP. Dans ce cas, on n'utilisera plus seulement un filtrage de paquets, mais également un filtrage sur le protocole lui-même, ce qui permet, par exemple, de faire du "contrôle parental" ou du contrôle de trafic web tout court.

Face à cet avantage, il y a pas mal d'inconvénients, dus à tous les effets pervers des fonctionnalités que l'on peut ajouter à un proxy. Parmi les inconvénients les plus graves:

- La durée de vie du cache peut être mal paramétrée, la vérification de validité du contenu également et le proxy peut fournir des informations qui ne sont plus à jour.
- Les proxys ont souvent des fonctions de restriction d'accès qui peuvent aboutir à un régime franchement totalitaire. (Contrôle parental chez AOL, par exemple, mais ce n'est pas forcément vous qui choisissez les filtrages).
- Les fonctions de traçage ne manquent pas non plus, ce qui permet d'espionner de façon très efficace ce que les utilisateurs de votre réseau font sur le Net.

Normalement, le navigateur Internet doit être paramétré pour utiliser un serveur proxy (outils/Options Internet/Connexions/Paramètres LAN dans Internet Explorer). Si l'installation est faite proprement, l'utilisateur devrait pouvoir choisir d'utiliser le proxy ou non. Souvent cependant, l'administrateur du réseau va bloquer le passage direct sur le port 80, obligeant les utilisateurs à passer par le proxy. Là encore, au moins, les utilisateurs sont avertis.

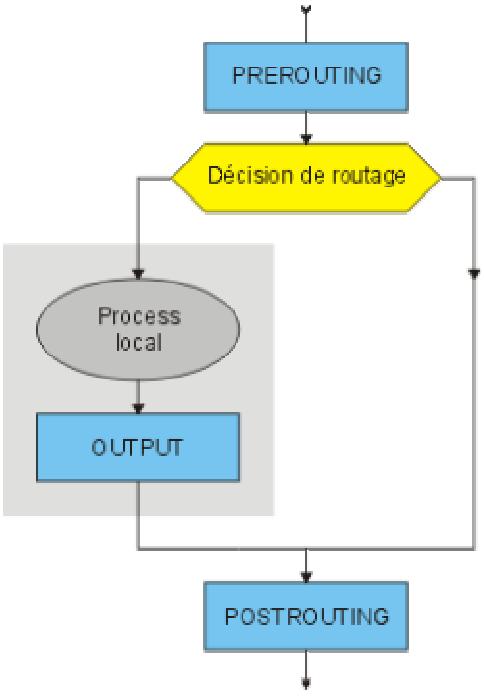
Le proxy transparent est beaucoup plus pernicieux, parce qu'il ne nécessite aucun paramétrage du navigateur et l'utilisateur ne sait pas qu'il passe par un proxy.

Le principe est simple, il suffit de rediriger tous les paquets dont le port de destination est 80 vers le proxy transparent, qui peut être placé sur la passerelle elle même. Ceux qui disposent d'une passerelle Linux peuvent assez facilement monter la manip en installant le proxy SQUID (configuré convenablement pour faire un proxy transparent) et en utilisant iptables pour faire de la redirection sur le service squid local.

Si vous voulez plus de détails sur ces pratiques qui flirtent avec le douteux, visitez le chapitre consacré à HTTP.

Ce ne sont pas les seules manipulations possibles, mais ce sont celles qui paraissent les plus souvent utilisées.

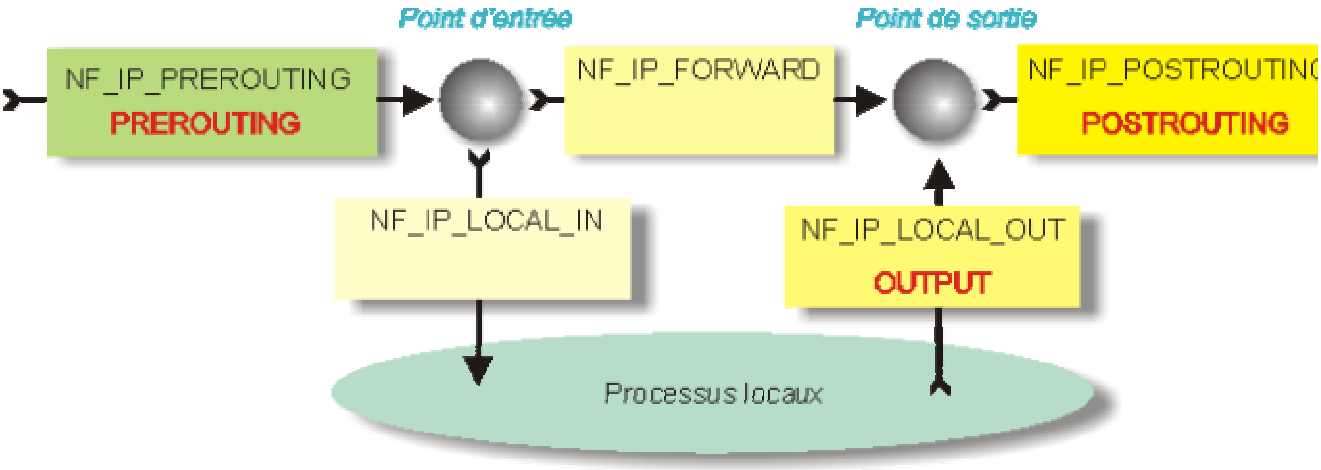
Et comment ça marche ?



La table NAT est organisée comme ci-contre:

- Comme son nom l'indique, la chaîne PREROUTING va bricoler les "sockets" avant les décisions de routage. Nous nous en servirons pour faire du DNAT (Destination NAT), autrement dit, pour modifier la "socket" du destinataire.
- La chaîne POSTROUTING intervient à la sortie du routeur. Elle servira à faire du SNAT (Source NAT) dont par exemple, le masquage d'adresse.
- La chaîne OUTPUT, quant-à elle, permet de modifier le socket de destination d'un paquet issu d'un processus local. L'utilité de cette chaîne n'est pas évidente, dans la mesure où, normalement, les paquets sortant d'un processus local devraient aussi passer par POSTROUTING.
  La seule possibilité supplémentaire est de pouvoir rediriger les paquets sortent d'un processus local à destination d'une cible extérieure, vers un autre processus local (127.0.0.1).

Là encore, nous pouvons l'illustrer de façon différente :



Les possibilités offertes par le NAT sont quasiment infinies. Nous avons vu les plus fréquentes :

- Masquage d'adresse, pour permettre à tout un réseau privé d'accéder au Net lorsque l'on ne dispose que d'une seule adresse IP valide sur le Net,
- redirection d'un service serveur adressé sur la passerelle vers un serveur situé dans le réseau privé, ça peut être utile pour les joueurs en réseau, mais aussi pour des applications plus professionnelles.

Pour que tout ça fonctionne correctement, le système s'appuie sur le suivi de connexion. Nous pouvons donc nous attendre à trouver des modules spécialisés pour certains

protocoles, dont le FTP, toujours lui. Ainsi, le module ip_nat_ftp sera nécessaire si vous voulez travailler proprement en FTP.

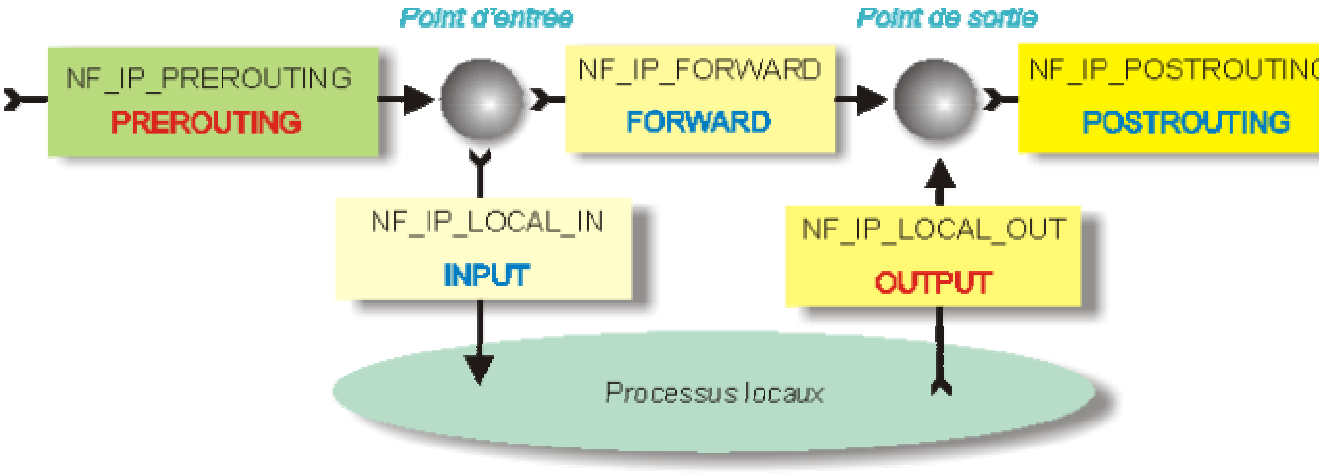
Et le reste

Ce n'est pas tout...

Netfilter permet encore d'autres choses, qui sortent plus ou moins du cadre de cet exposé (peut-être un jour ?).

Mangle.

Nous n'avons pas parlé de la table Mangle. Cette table permet d'effectuer un marquage des paquets.



Nous pouvions, avec les premières versions de Netfilter, marquer les paquets en PREROUTING ou en OUTPUT pour les sorties d'un processus local (en rouge sur l'illustration). Notez que depuis la version 2.4.18 du noyau, le système a été étendu à tous les "hooks" (en bleu sur l'illustration).

L'intérêt de ce marquage, qui n'est visible que dans la pile de la machine, est de pouvoir être relu par d'autres fonctions comme IPROUTE ou la gestion de la qualité de service (QoS).

Ainsi, nous pouvons disposer de toute la puissance de Netfilter pour la sélection de divers types de paquets, et utiliser ensuite ce marquage pour le routage ou les priorités de passage.

Donner ici des exemples précis nous mènerait trop loin parce qu'il faudrait étudier en détail IProute2 et les fonctions de QoS des noyaux 2.4 et la vie est courte.

Voici tout de même un cas de figure qui serait gérable par ce système :

- Nous disposons de deux liens sur le Net :

    - L'un, très rapide et très fiable, mais très cher et facturé au volume de données,
    - L'autre, classique, comme une connexion ADSL, avec les limites que nous leurs connaissons.
- Nous souhaitons exploiter au mieux ces deux connexions, par exemple de la façon suivante :
    - Nous devons mettre à jour le contenu d'un serveur distant. Il faut le faire de façon rapide et sûre. Il n'y a pas forcément beaucoup de données à transmettre, mais il est impératif que ce soit fait le plus rapidement et le plus sûrement possible.
    - Nous devons assurer un accès au Net pour les utilisateurs du réseau local, mais avec une qualité de service plus faible.

Avec le marquage de paquets associé à IProute, nous pourrons arriver à faire passer les mises à jour du serveur sur le lien rapide mais cher et tout le reste sur le lien ADSL.

- Nous disposons d'une connexion ADSL et il arrive très souvent que certains utilisateurs fassent du téléchargement FTP sur des serveurs rapides. Chaque fois qu'un téléchargement est lancé, toute la bande passante Download est utilisée et les autres utilisateurs ne peuvent plus surfer dans de bonnes conditions.
- En exploitant le marquage de paquets associé aux fonctions de QoS, nous pourrons restreindre la bande passante exploitée par le download FTP afin de laisser un peu d'espace pour les autres activités.
- Ceci peut aussi être appliqué aux transferts "peer-to-peer", qui ont l'inconvénient de monopoliser le peu de bande passante upload dont on dispose sur des connexions asymétriques comme ADSL ou câble.

Les logs.

Netfilter propose un système de log puissant. Il ne s'agit pas ici d'une table, mais d'une cible. Nous verrons plus en détail dans la suite ce que sont les cibles, disons pour le moment qu'il en existe deux qui sont particulières.

Le cible LOG.

Elle permet de remonter vers le démon syslog, avec, par défaut, le niveau "warning", des messages décrivant les paquets qui satisfont à la règle qui pointe vers LOG. Dans une distribution Mandrake, nous retrouverons donc leur trace dans /var/log/messages.

Juste un exemple trivial, pour voir ce que ça donne :

iptables -A INPUT -i eth0 -p icmp -j LOG

Ce qui veut dire en français :

Ajouter à la chaîne INPUT la règle suivante : envoyer vers la cible LOG tout paquet ICMP qui entre par eth0

La machine dont l'IP de eth0 est 192.168.0.253 envoie un ping sur la machine192.168.0.251.

ping -n 1 192.168.0.251

Nous allons tracer la réponse au ping (INPUT). Nous récupérons cette trace dans /var/log/messages :

```
Dec  1 22:40:11 linux kernel:
        IN=eth0
        OUT= MAC=00:00:b4:bb:5d:ee:00:20:18:29:11:31:08:00
        SRC=192.168.0.251
        DST=192.168.0.253
        LEN=84 TOS=0x00
        PREC=0x00
        TTL=255 ID=4938
        PROTO=ICMP TYPE=0
        CODE=0
        ID=53771
        SEQ=256
```

Ce qui est surligné en jaune correspond à la machine qui trace, c'est à dire celle qui envoie le ping et attend la réponse (qui est tracée). Ce qui est surligné en vert correspond à la cible du ping qui répond.

Comme vous le voyez, c'est un bon moyen pour se faire de la lecture facile, propre à vous aider en cas d'insomnie. Si la commande ping est écrite de la façon suivante :

```
ping -n 100 192.168.0.251
```

Nous génèrerons 100 fois une trace similaire à celle que nous venons de voir. Compter les moutons qui sautent la barrière est sans doute aussi efficace.

Pour éviter que les logs n'arrivent à remplir votre disque dur, il existe la directive "limit" qui permet, comme son nom l'indique, de limiter l'envoi vers la cible de paquets satisfaisant à la règle. Cette directive à elle seule mériterait toute une page d'explications.

La cible ULOG.

Plutôt que d'utiliser syslog, cette cible permet d'envoyer les paquets à un démon spécialisé : ulogd. Ce démon permet d'obtenir des logs plus présentables, voir stockés dans une base de données comme MySQL. Ce démon n'est malheureusement pas fourni dans la distribution Mandrake 9

IPtables

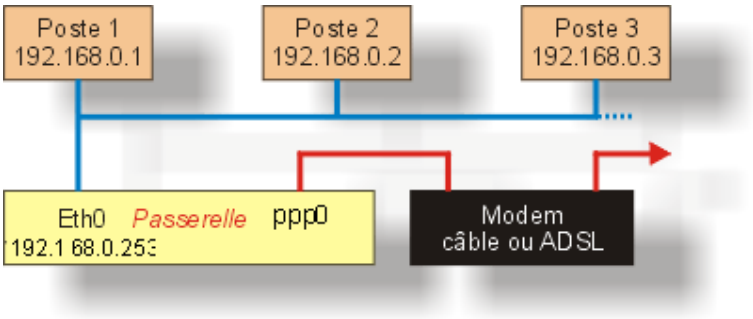
Encore une fois, il n'est pas question de reprendre toute la documentation d'IPtables. Nous allons simplement examiner quelques règles simples d'un usage courant. Pour étudier la syntaxe, consultez:

- Les pages man (man iptables)
- Rusty's Remarkably Unreliable Guides

Les pages de man, bien que pas toujours très agréables à lire, sont essentielles, pour la simple raison que Netfilter/IPtables sont des outils en pleine évolution et que d'une version à l'autre, de nouvelles fonctionnalités peuvent apparaître.

Manipulations diverses.

Pour ce qui suit, nous allons faire de la pratique. Mandrake 9 montée en passerelle, tel que décrit dans le chapitre "un routeur Linux".



Rappelons-le, il s'agit de masquer tous les clients d'un réseau privé (par exemple 192.168.0.0) derrière l'unique adresse officielle attribuée par le fournisseur d'accès, et d'assurer un minimum de sécurité sur la totalité de l'installation.

Initialisation des tables.

Pour commencer, nous allons tout fermer au niveau de la passerelle dans la table "filter".

```
# Nous vidons les chaînes :
iptables -F
# Nous supprimons d'éventuelles chaînes personnelles :
iptables -X

# Nous les faisons pointer par défaut sur DROP

iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
# Nous faisons de même avec toutes les autres tables,
# à savoir "nat" et "mangle", mais en les faisant pointer
# par défaut sur ACCEPT. Ca ne pose pas de problèmes
# puisque tout est bloqué au niveau "filter"

iptables -t nat -F
iptables -t nat -X
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT
iptables -t nat -P OUTPUT ACCEPT
iptables -t mangle -F
iptables -t mangle -X
iptables -t mangle -P PREROUTING ACCEPT
iptables -t mangle -P INPUT ACCEPT
iptables -t mangle -P FORWARD ACCEPT
iptables -t mangle -P POSTROUTING ACCEPT
```

Notez que dans tout ça, on n'a pas changé grand chose par rapport à l'état de ces tables tel qu'on le trouve après un boot de la machine, sans modifications particulières des scripts de démarrage, hormis la cible par défaut des règles de la table "filter" que l'on a passé à DROP. Tout de même, cette manipulation préliminaire a eu pour conséquences :

- Que l'on est parfaitement sûr de l'état de Netfilter,
- Que l'on a commencé à se familiariser un peu avec le langage IPtables.

Où en sommes-nous ?

Normalement, plus rien ne doit passer nulle part. Essayez des pings dans tous les sens, entre la passerelle et votre réseau privé ou vers le Net, entre un poste de votre réseau privé et la passerelle, rien ne devrait passer.

Ouvrons quelques portes.

```
# Nous considérons que la machine elle même est sûre
# et que les processus locaux peuvent communiquer entre eux
# via l'interface locale :

iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Nous considérons que notre réseau local est
# également sûr (ce qui n'est pas forcément vrai, d'ailleurs).

iptables -A INPUT -i eth0 -j ACCEPT
iptables -A OUTPUT -o eth0 -j ACCEPT
```

A ce stade, nous avons la situation suivante :

- Si, sur votre passerelle, vous faites ping 127.0.0.1, ça répond,
- si, sur votre passerelle, vous faites un ping sur un hôte du réseau privé, ça répond,
- si, sur un hôte du réseau privé, vous faites un ping sur la passerelle, ça répond aussi.

Mais...

- Si, depuis la passerelle, on fait des pings n'importe où sur le Net, ça ne répondra pas, à cause du DROP sur OUTPUT par défaut,
- si, depuis n'importe où sur le Net, on fait des pings sur votre passerelle, ça ne répondra pas non plus, à cause du DROP sur INPUT par défaut,
- si, depuis un hôte de votre réseau, on fait un ping n'importe où sur le Net, ça ne répondra encore pas, pour deux raisons :
    - Nous n'avons pas placé de règle de NAT entre le réseau local et le Net,
    - FORWARD fait DROP sur tout ce qui passe.

Faisons maintenant du NAT :

```
# Translation d'adresses pour tout ce qui traverse la passerelle
# en sortant par ppp0
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

Nous pourrions ici restreindre le NAT à une plage d'IPs du réseau local :

```
iptables -t nat -A POSTROUTING -s 192.168.0.0/255.255.255.0 -o ppp0 -j MASQUERADE
```

Ou même à une liste d'IP bien définies :

```
iptables -t nat -A POSTROUTING -s 192.168.0.10 -o ppp0 -j MASQUERADE
iptables -t nat -A POSTROUTING -s 192.168.0.11 -o ppp0 -j MASQUERADE
```

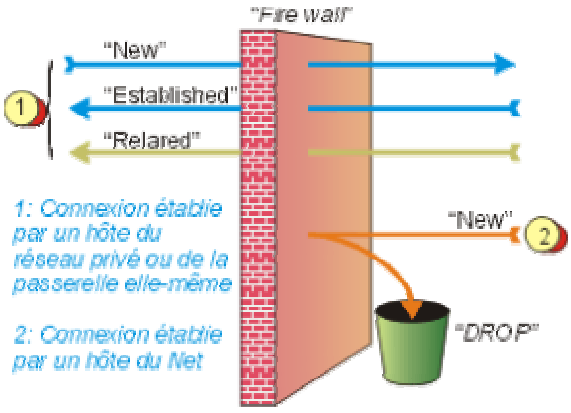
Ceci peut être utile, surtout en sachant que vous pouvez détruire une règle et une seule :

```
iptables -t nat -D POSTROUTING -s 192.168.0.11 -o ppp0 -j MASQUERADE
```

Ainsi, si vous avez, par exemple, des enfants qui usent de votre connexion permanente de façon un peu trop permanente, vous pourrez aisément, avec l'aide du démon "cron" n'accorder l'accès au Net que pour certaines plages horaires.

Mais ça ne suffit pas encore pour fonctionner, les pings depuis le réseau local vers le Net ne passent toujours pas. Normal, FORWARD fait toujours DROP sur tout ce qui passe. Nous devons accorder des autorisations de passage sur FORWARD.

Utilisation de conntrack.



Le suivi de connexion est intéressant, bien qu'il consomme un peu plus de ressources sur votre passerelle. Nous l'avons vu, son avantage est qu'il permet d'obtenir des informations sur toute connexion en cours. Ces informations sont principalement:

- **NEW**
  Une nouvelle connexion est établie.
- **ESTABLISHED**
  La connexion analysée a déjà été établie
- **RELATED**
  La connexion est en relation avec une autre connexion déjà établie (par exemple, dans le FTP actif).
- **INVALID**
  Le paquet n'appartient à aucune des trois catégories précédentes.

Il est possible, par exemple, de n'accepter dans les deux sens (vers et depuis l'Internet) que les connections déjà établies ou en relation avec des connexions déjà établies et de n'accepter des nouvelles connexions que depuis notre installation vers l'Internet. De cette manière, notre réseau local pourra se connecter sur tout serveur Internet, mais aucune nouvelle connexion ne pourra être créée depuis le Net vers notre installation.

En français, nous allons faire :

- Toutes les connexions : **nouvelles**, **établies** et **associées** à une connexion établie qui entrent par eth0 (réseau local) et veulent sortir par ppp0 (le Net), peuvent passer,
- Toutes les connexions : **établies** et **associées** à une connexion établie qui entrent par ppp0 et veulent aller vers eth0 pourront également passer.

En langage iptables :

```
# Toutes les connexions qui sortent du LAN vers le Net
# sont acceptées
```

```
iptables -A FORWARD -i eth0 -o ppp0 -m state --state NEW,ESTABLISHED,RELATED -j
ACCEPT
# Nous autions aussi bien pu écrire :
# iptables -A FORWARD -i eth0 -o ppp0 -m state --state ! INVALID -j ACCEPT

# Seules les connexions déjà établies ou en relation avec
# des connexions établies sont acceptées venant du Net vers le LAN
iptables -A FORWARD -i ppp0 -o eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
```

ping http://www.grenouille.com/ (par exemple) se met à fonctionner.

Le chapeau magique.

Maintenant que nous avons mis conntrack en oeuvre, nous aimerions bien voir un peu comment il opère... Il se trouve qu'il est possible d'observer la table de suivi de connexions qui se présente sous la forme d'un fichier virtuel en /proc/net/ip_conntrack.

Pour interpréter plus facilement ce qu'il suit, il faut savoir plusieurs choses :

- Un client du LAN dispose de l'IP 192.168.0.10,
- la passerelle dispose de l'IP 192.168.0.253 sur le LAN
- la passerelle dispose de l'IP 80.8.130.97 sur le Net
- le client navigue sur le serveur HTTP d'IP 213.186.35.33

```
tcp 6 14 CLOSE_WAIT
        src=192.168.0.10 dst=213.186.35.33 sport=1102 dport=80
        src=213.186.35.33 dst=80.8.130.97 sport=80 dport=1102 [ASSURED] use=1

tcp 6 431991 ESTABLISHED
        src=192.168.0.10 dst=213.186.35.33 sport=1103 dport=80
        src=213.186.35.33 dst=80.8.130.97 sport=80 dport=1103 [ASSURED] use=1

tcp 6 73 TIME_WAIT
        src=192.168.0.10 dst=213.186.35.33 sport=1104 dport=80
        src=213.186.35.33 dst=80.8.130.97 sport=80 dport=1104 [ASSURED] use=1

tcp 6 82 SYN_SENT
        src=192.168.0.10 dst=213.186.35.33 sport=1105 dport=80 [UNREPLIED]
        src=213.186.35.33 dst=80.8.130.97 sport=80 dport=1105 use=1

tcp 6 51 CLOSE_WAIT
        src=192.168.0.10 dst=213.186.35.33 sport=1106 dport=80
        src=213.186.35.33 dst=80.8.130.97 sport=80 dport=1106 [ASSURED] use=1
```

Le troisième champ est un timer, l'entrée est effacée de la table lorsque le timer tombe à zéro. Bien entendu, cette table est en mémoire, ce n'est pas un vrai fichier, son contenue évolue donc perpétuellement au cours du temps.

- Vous ne pourrez pas la visualiser efficacement son contenu avec l'outil "tail",
- cette table prend de la place en mémoire, d'autant plus que le nombre de clients sur le LAN est important et leur activité grande. Sur un petit réseau domestique, ce sera rarement un problème, mais sur un réseau d'entreprise, il faudra rester attentif à la mémoire disponible.

Où en somme nous ?

- Depuis le réseau local, nous accédons à la passerelle,
- depuis le réseau local, nous accédons au Net,
- depuis la passerelle, nous accédons au réseau local,
- depuis la passerelle, nous n'accédons pas au Net
- depuis le Net, nous n'accédons pas à la passerelle (nous avons toujours un DROP en INPUT sur ppp0)
- depuis le net, nous ne pouvons accéder aux hôtes du réseau local que sur des connexions qu'ils ont établies et dont ils sont clients. Autrement dit, aucun serveur, éventuellement placé sur le réseau local ne sera accessible depuis le Net

Amélioration possibles.

Un DNS local.

Si vous installez sur votre passerelle un serveur DNS, il faudra qu'il puisse envoyer ses requêtes sur le Net, ce qui n'est actuellement pas possible. Il faut ouvrir une voie en UDP conforme aux besoins d'une requête DNS :

- Votre serveur envoie une requête UDP à destination du port 53 d'un serveur DNS
- il attend une réponse sur un port supérieur ou égal à 1025, venant d'un port 53.

En langage IPtables :

```
# Autorisation des requêtes DNS locales
iptables -A OUTPUT -o ppp0 -p udp --sport 1024: --dport 53 -m state --state ! INVALID -j ACCEPT
iptables -A INPUT -i ppp0 -p udp --sport 53 --dport 1024: -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Voilà des syntaxes qui commencent à être sympathiques...

Lorsque l'on veut définir, non pas un port mais une plage de ports, les écritures suivantes sont autorisées :

- --dport 1024:1999 autorisera la plage de ports [1024,1999] en destination (ça marche aussi avec --sport),
- -- dport 1024: veut dire que tous les ports supérieurs où égaux à 1024 seront acceptés en destination.

Un SMTP local :

Vous pouvez désirer implanter un SMTP sur votre passerelle, pour envoyer votre courrier depuis le LAN sans vous préoccuper des disponibilités ou des lenteurs du SMTP de votre FAI. Voir le chapitre SMTP à ce sujet.

Deux options sont possibles :

- Votre SMTP enverra vos courrier directement aux destinataires,
- votre SMTP enverra tous vos courriers au SMTP de votre FAI qui effectuera lui-mêmes les livraisons.

La première option est la plus rapide. Malheureusement, à cause de nombreux débordements, la tendance actuelle consiste à refuser les messages provenant de serveurs SMTP non "officiels", c'est par exemple le cas actuellement entre Wanadoo et AOL. AOL refuse tout message en provenance de wanadoo.fr autre que ceux qui lui

arrivent des SMTP officiels de Wanadoo. Comme il est clair que cette tendance n'ira qu'en s'accentuant, il demeure plus sage d'utiliser la seconde méthode. Vous serez tributaire du bon fonctionnement du SMTP de votre FAI, mais ça ne se verra pas au niveau de vos clients du LAN. Ce sera votre SMTP local qui assurera les éventuelles attentes.

Sachant qu'un serveur SMTP écoute sur le port 25 et utilise tcp :

```
# Autorisation des envois SMTP locaux
iptables -A OUTPUT -o ppp0 -p tcp --sport 1024: --dport 25 -m state --state ! INVALID -j ACCEPT
iptables -A INPUT -i ppp0 -p tcp --sport 25 --dport 1024: -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Et vous pouvez même restreindre encore d'avantage, si vous avez adopté la seconde stratégie d'envoi :

```
# Autorisation des envois SMTP locaux vers le SMTP du FAI
iptables -A OUTPUT -o ppp0 -p tcp --sport 1024: -s smtp.wanadoo.fr --dport 25 -m state --state ! INVALID -j ACCEPT
iptables -A INPUT -i ppp0 -p tcp -d smtp.wanadoo.fr --sport 25 --dport 1024: -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Si, si, ça fonctionne, à condition bien entendu que votre passerelle soit en mesure de résoudre les noms au moment où vous écrivez la règle. Attention donc, si vous faites tout ça lors de l'initialisation de la machine, il faudra que :

- Les services réseaux soient montés,
- la connexion PPPoE établie,
- le service DNS démarré.

Vous pouvez, plus simplement, indiquer non pas le nom du serveur mais son IP.

Un accès SSH depuis le Net.

Vous pouvez souhaiter pouvoir accéder à votre passerelle depuis le Net, pour peu que vous ayez un moyen de connaître à tout moment son adresse IP. Sachant que ssh utilise tcp sur le port 22 :

```
# Accès SSH depuis le Net
iptables -A INPUT -p tcp --dport ssh -i ppp0 -j ACCEPT
iptables -A OUTPUT -p tcp --sport ssh -o ppp0 -j ACCEPT
```

Nous pouvons effectivement définir un port par le service qui lui est normalement associé (ici ssh).

Si vous devez accéder à votre passerelle depuis une IP fixe sur le Net, vous pouvez largement restreindre cette règle en n'acceptant les connexions ssh que depuis et vers cette IP.

ICMP, c'est parfois utile.

Le protocole ICMP, même s'il présente quelques dangers, rend tout de même quelques services appréciables, dans le cas d'erreurs de transmission et aussi dans la découverte du MTU.

Il se trouve que, pour les messages d'erreur ICMP, l'en-tête du paquet qui a généré l'erreur est reproduite dans le message. Le suivi de connexion ICMP s'en sert pour déclarer ce paquet "RELATED".

En ce qui concerne les clients du LAN, avec les règles que nous avons écrites, il ne devrait pas y avoir de problèmes, puis qu'on laisse passer tout paquet "RELATED" sans distinction de protocole.

En revanche, pour la passerelle elle-même, si l'on a activé SMTP et DNS, il peut s'avérer intéressant d'ajouter la ligne :

iptables -A INPUT -p icmp -m state --state RELATED -j ACCEPT

De cette manière, une erreur ICMP passera, mais votre passerelle ne répondra pas aux pings, ni à aucune autre interrogation ICMP.

Encore une dernière recette.

Vous pouvez faire quelque chose de très simple, mais qui reste tout de même moins sûr.

```
# Table Filter (table par défaut).
#--------------------------------
#Vidage des chaînes
iptables -F
#Destruction des chaînes "personnelles"
iptables -X
```

Changement de stratégie par défaut: Nous n'acceptons plus rien

- ni en entrée
- ni en traversée de la passerelle

Mais nous acceptons tout ce qui sort (localement) de la passerelle

```
#Stratégie par défaut:
#INPUT et FORWARD sont DROP
iptables -P INPUT DROP
iptables -P FORWARD DROP
#OUTPUT est ACCEPT
iptables -P OUTPUT ACCEPT
# Init. des tables NAT et MANGLE:
#-------------------------------
iptables -t nat -F
iptables -t nat -X
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT
iptables -t nat -P OUTPUT ACCEPT
iptables -t mangle -F
iptables -t mangle -X
iptables -t mangle -P PREROUTING ACCEPT
iptables -t mangle -P OUTPUT ACCEPT
# Mise en place du NAT
#--------------------
iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o ppp0 -j MASQUERADE
```

Maintenant, nous allons créer une chaîne particulière, que nous allons appeler "SuiviConnexions" et qui va gérer ce suivi.

```
# Création d'une chaîne personnelle: "SuiviConnexions"
iptables -N SuiviConnexions
# Filtrage de suivi dans cette chaîne:
# Seules les nouvelles connexions qui ne viennent pas du Net sont acceptées
iptables -A SuiviConnexions -m state --state NEW -i ! ppp0 -j ACCEPT
```

Ca veut dire, plus clairement, toutes les connexions qui n'entrent pas par ppp0, c'est à dire dans notre cas, qui entrent par eth0, mais aussi par l'interface locale (lo)

```
# Toutes les connexions établies et relatives sont acceptées
iptables -A SuiviConnexions -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Cette chaîne va maintenant servir de cible commune pour les deux chaînes standard INPUT et FORWARD

```
# Et les deux chaînes INPUT et FORWARD pointent sur SuiviConnexions
iptables -A INPUT -j SuiviConnexions
iptables -A FORWARD -j SuiviConnexions
```

Et le tour est joué, puisque OUTPUT accepte tout. Il faut faire beaucoup confiance au bon fonctionnement de conntrack et ne pas trop se poser de questions sur les coups tordus qui peuvent arriver à passer quand même là dedans, mais au premier coup d'oeil, votre machine paraîtra invisible sur le Net.

Enfin, pour ssh :

```
# Accès SSH depuis le Net
iptables -A INPUT -p tcp --dport ssh -j ACCEPT
```

Le FTP actif.

Très souvent, les clients du LAN derrière un routeur NAT ne peuvent accéder à des serveurs FTP sur le Net qu'en mode passif. Sans précautions particulières, ce sera également le cas ici.

Cependant, Netfilter permet de s'affranchir de cette limitation, en exploitant les modules spécialisés, ip_nat_ftp et ip_conntrack_ftp. Ceci nous amène à dire quelques mots du chargement de ces modules, à propos desquels nous ne nous sommes pas beaucoup posé de questions.

Netfilter est capable de charger dynamiquement la plupart des modules qui lui sont nécessaires, en fonction des règles écrites. Faites un "lsmod" et voyez les modules impliqués dans Netfilter (ils sont tous dans /lib/modules/2.4.19-16mdk/kernel/net/ipv4/netfilter, pour la Mandrake 9).

Cependant, les modules nécessaires au FTP actif ne se chargent pas automatiquement. Vous pourrez les monter avec modprobe pour faire les tests. Si vous voulez les rendre automatiquement disponibles à chaque démarrage, référencez-les par exemple dans le fichier /etc/modules.

Pour conserver tout ce beau travail.

C'est le moment d'utiliser un outil fourni avec iptables : le script iptables-save. Ce script envoie sur le flux de sortie par défaut, normalement l'écran, le contenu des chaînes de toutes les tables, dans un format relativement lisible pour l'être humain :

```
[root@linux root]# iptables-save
# Generated by iptables-save v1.2.6a on Mon Dec  2 18:22:31 2002
*mangle
:PREROUTING ACCEPT [4110:906437]
:INPUT ACCEPT [113562:22595477]
:FORWARD ACCEPT [2160:709057]
:OUTPUT ACCEPT [2308:208513]
:POSTROUTING ACCEPT [68746:14733820]
COMMIT
# Completed on Mon Dec  2 18:22:31 2002
# Generated by iptables-save v1.2.6a on Mon Dec  2 18:22:31 2002
*filter
:INPUT DROP [392:38736]
:FORWARD DROP [20:944]
:OUTPUT DROP [616:25100]
-A INPUT -i lo -j ACCEPT
-A INPUT -i eth0 -j ACCEPT
-A FORWARD -i eth0 -o ppp0 -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i ppp0 -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -o eth0 -j ACCEPT
COMMIT
# Completed on Mon Dec  2 18:22:31 2002
# Generated by iptables-save v1.2.6a on Mon Dec  2 18:22:31 2002
*nat
:PREROUTING ACCEPT [781:63004]
:POSTROUTING ACCEPT [331:18116]
:OUTPUT ACCEPT [1028:49802]
-A POSTROUTING -o ppp0 -j MASQUERADE
COMMIT
# Completed on Mon Dec  2 18:22:31 2002
```

Nous retrouvons bien là dedans tout ce que nous avons défini plus haut. Mais cette commande a un autre avantage. En dirigeant sa sortie vers un fichier, vous obtenez un fichier de configuration qui sera exploitable par un autre script : iptables-restore

En d'autres termes, si vous faites :

```
iptables-save  > /root/maconfig.iptables
```

vous pourrez refaire ensuite :

```
iptable-restore < /root/maconfig.iptables
```

Pour restaurer intégralement votre configuration.

A noter, toujours sur les distributions Mandrake, un script de démarrage /etc/init.d/iptables qui permet plusieurs choses :

- /etc/init.d/iptables save
  permet de sauvegarder l'état des tables dans /etc/sysconfig/iptables (en utilisant iptables-save, comme vu plus haut),

- /etc/init.d/iptables start (ou restart)
  permet de restaurer l'état des tables au moyen du fichier /etc/sysconfig/iptables
- /etc/init.d/iptables stop
  efface toutes vos règles et met par défaut ACCEPT sur toutes les chaînes de toutes les tables
- /etc/init.d/iptables panic
  votre machine se ferme comme une huitre.

Vous pouvez donc facilement user de ce script pour sauvegarder vos règles et les restaurer à chaque redémarrage.

Conclusions.

Beaucoup de choses n'ont pas été dites, Netfilter nécessiterait un livre entier. Parmi ce dont nous n'avons pas parlé et qui peut s'avérer utile, même pour un réseau domestique :

- Comment palcer, par exemple, un serveur HTTP sur le LAN et s'arranger pour qu'il soit visible depuis le Net ? C'est possible en utilisant du NAT de destination (PREROUTING). Je vous laisse chercher...
- Nous n'avons que survolé l'utilisation de la cible LOG, pour tracer des paquets indésirables, ou simplement pour déverminer des règles qui ne fonctionnent pas quomme on le souhaiterait. La cible LOG est un peu spéciale, dans la mesure où elle n'affecte pas la destination du paquet tracé. Ainsi, vous devrez écrire des règles pour les logs en plus de votre filtrage et de votre NAT.
- L'utilisation de la table mangle conjointement avec le QoS pour exploiter au mieux votre bande passante.

Je ne suis pas un expert en sécurité. Ne comptez pas sur moi pour vous donner toutes les ficelles de sécurisation de votre installation. D'ailleurs, rien n'est définitif dans ce domaine.

D'autres sources d'informations.

Reprenons quelques liens qui vous en apprendront d'avantage :

- Rusty's Remarkably Unreliable Guides
  Vous y trouverez plusieurs documents, dont certains en français, sur le filtrage, le NAT et même l'architecture interne de Netfilter, sous forme de HOWTO,
- Le site officiel de Netfilter
  vous y trouverez absolument toutes les docs possibles, la plupart du temps en anglais, mais un bon nombre de HOWTOS est disponibles en plusieurs langues, dont le français
- Le Guide d'installation et de configuration de Linux
  propose un chapitre sur les firewalls et le partage de connexion avec Netfilter
- Un script qui peut être un bon exercice d'apprentissage du langage iptables,
- IPTables par l'exemple
  Cet article présente de façon pratique la mise en place d'un firewall avec IPTables (en français).

Et probablement beaucoup d'autres que je ne connais même pas :)