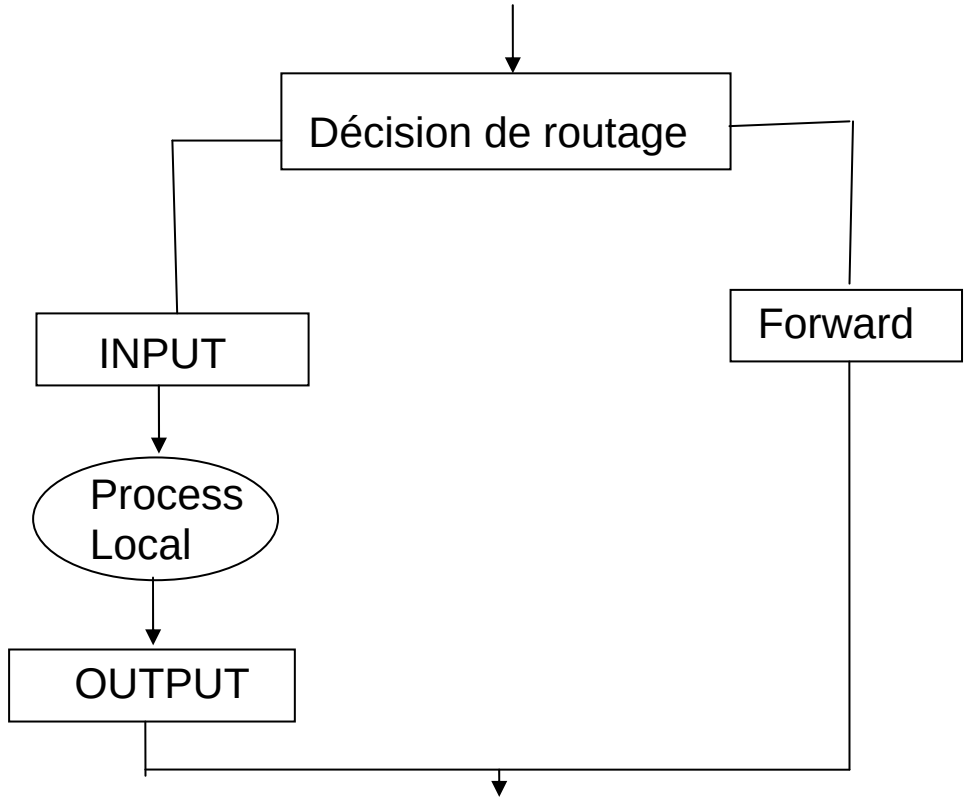
TP4 : Firewall NETFILTER

IPTables est un outil Linux pour gérer Ie pare-feu qui est intégré au noyau Linux 2.4 (et supérieur). L'architecture du noyau pour Ie système de pare-feu s'appelle 'netfilter'. Un pare feu est un système qui filtre les connexions réseaux. Cela permet d'interdire des connexions sur certains ports d'un ordinateur, de limiter le nombre de connexions, de limiter les bandes passantes etc. Bref de gérer de nombreux aspects des flux réseaux. Netfilter a de très nombreuses fonctionnalités, nous en étudierons les parties essentielles (filtrage de paquets, suivi de connexions, NAT[1]. ...).

Le principe de fonctionnement est simple, lorsque la carte réseau reçoit un paquet celui-ci est transmis à la partie netfilter du noyau. Netfilter va ensuite étudier ce paquet (les entêtes et le contenu) et en se basant sur des règles que l'administrateur aura défini, il va choisir de laisser passer le paquet intact, de modifier ce paquet, de le transmettre à une autre machine ou encore d'interdire le passage.

Netfilter fonctionne en utilisant 3 tables invariables : filter, nat et mangle. La table filter correspond aux notions de filtrage réseaux (accepter/refuser un paquet}, la table nat correspond à des fonctions de routage (pour monter une passerelle par exemple) et la table mangle est utilisée pour modifier les paquets (par exemple pour faire du QoS[2]). Netfilter se base sur 3 listes de règles pour définir, son comportement vis-à-vis d'un paquet (au niveau de la table filter) . Voici un schéma qui récapitule le trajet virtuel d'un paquet au travers de ces 3 « chaînes » :



[1] Network Address Translation, consiste à modifier un paquet pou r en changer l'adresse IP source (SNAT: Source NAT) et/ou destination (DNAT : Dest NAT).

[2] Quality Of Service, consiste à donner des priorités en bande passante pour certains flux (utilises par exemple pour des applications de video conferences).

Quand un paquet arrive (par la carte réseau par exemple), le noyau regarde la destination de ce *paquet : c'est appelé le* routage. Si ce *paquet est destiné à cette machine, le paquet passe dans la chaîne INPUT. S'il la passe, les processus qui attendent le paquet le recevront. Autrement, si le noyau n'a pas le forwarding autorisé (machine qui n'est pas configurée en routeur), ou qu'il ne sait pas comment forwarder le paquet; le paquet est effacé. Si le forwarding est autorisé et que le paquet est destiné à un autre réseau, le paquet va directement à la chaîne FORWARD. S'il est accepté, il sera envoyé vers Ie réseau de·destination.*
La chaîne OUTPUT concerne les paquets qui ont été créé par la machine locale. Ces paquets passeront par la chaîne OUTPUT immédiatement. Chacune de ces chaînes peut donner plusieurs réponses possibles pour chaque paquet :
ACCEPT : le paquet est accepté
DROP : le paquet est refusé, on l'efface et *on ne répond rien*
REJET : le paquet est refusé et *on signale Ie rejet à l'expéditeur*

Chaque chaîne possède une règle par défaut « *policy » qui définit si l'on accepte ou refuse les paquets* « *par défaut* ». *Par défaut, les policy sont à ACCEPT.*

Nous verrons par la suite que I'administrateur peut créer des chaînes personnalisées en plus des 3 *chaînes par défaut qui lui permettent d'organiser ses règles de filtrage de manière plus propre* et *optimisée (lorsqu'on a plusieurs milliers de règles. c'est vraiment très utile).*

1) Prise en Main

Sur vos distributions Fedora , vous avez pu constater lors des précédents TP que Ie firewall était active par défaut, Ie package iptables et lui aussi installé ce que vous pouvez vérifier par la commande :

[root@fedora -] # yum list installed iptables (sous fedora install)
Installed Packages
iptables.i386
installed

Toujours en ligne de commande nous allons afficher la politique actuellement Appliquée à l'aide d'iptables:

[root @fedora -] # iptables -L – v

L'option -L liste les règles et -v demande Ie mode 'verbeux', faîtes un man iptables pour obtenir la liste détaillée des options.

Vous voyez donc pour chaque chaîne standard, la « policy» (comportement par défaut), Ie nombre de paquets qui a traversé les règles de la chaîne et le nombre d'octets correspondants et I'ensemble des règles actives (par défaut iptables utilise la table 'filter ' si on ne lui spécifie par une autre table via l'option '-t nom_de_table').

Vous allez consultez Ie manuel iptables pour effacer la chaîne non standard et Remettre à zéro toutes les règles afin que netfilter accepte tous les paquets (Ie firewall est ainsi en mode passif) .

=>Quelle est la commande pour effacer une règle d'une chaîne ?
=>Comment effacer toutes les règles d'une chaîne ?
=> Comment supprimer une chaîne ?

=> Effectuez et notez les étapes nécessaires à la suppression , de la quatrième chaîne sous les systèmes Fedora (et autres distributions RedHat), les règles de firewall appliquées au démarrage du service sont stockées dans Ie fichier /etc/sysconfig/iptables, examinez ce fichier, que retrouvez-vous ?

=> Listez les options du service iptables en entrant:

[r oot@f edora -] #service iptables

Executez la commande permettant de sauvegarder la configuration actuelle (quelle est-elle ?), que constatez-vous quant au fichier cité plus haut ? Vérifiez en redémarrant Ie service.

2) Filtrage de ports

Nous allons maintenant interdire les connexions sur Ie port 22 (ssh), pour cela, il faut interdire les paquets dans la chaîne INPUT sur Ie port 22 :

root# iptables -A INPUT - p tcp - - dpor t 22 -j DROP

Cette commande signifie :

-A INPUT => ajoute en bas de la liste des règles de la chaîne INPUT
-p tcp => pour les paquets qui utilisent Ie protocole TCP
--dport => pour les paquets qui sont à destination du port 22
-j DROP => l'action : DROP les paquets (on efface les paquets), Ie -j peut

également servir à rediriger sur une autre chaîne et/ou à effectuer différentes actions (ef. Ie man iptables, section TARGET EXTENSIONS).

Demandez maintenant à votre voisin d'essayer de se connecter en ssh sur votre machine.
ici nous avons utilisé DROP, c'est-à-dire qu'on efface Ie paquet mais qu 'on ne dit rien à l'expéditeur. Le résultat comme vous Ie voyez, c'est que la connexion ssh depuis Ie client n'est plus possible, mais également Ie client ssh met un certain temps à vous signaler Ie problème. II attend désespérément une réponse du serveur) Si on avait utilisé REJECT au lieu de DROP, l'erreur serait instantanée. Refaites la commande qui liste les règles de filtrage :

root# iptables -L -v

Cela doit donner quelque chose comme :
.
Chain INPUT (policy ACCEPT 57153 packets , 7029K bytes)

pkts bytes target prot opt in out source destination
14 1024 DROP tcp - - * * 0 .0 .0 . 0/ 0 0 .0 .0 .0/0 tcp dpt : 22

Nous allons maintenant autoriser uniquement votre voisin à se connecter sur votre machine en ssh. Pour cela, nous allons insérer une règle pour votre voisin avant la règle qui DROP les paquets.
II est important de la placer avant, car les règles sont exécutées dans l'ordre ou elles ont été placées dans la liste, donc si la première règle rejette Ie paquet, celui-ci n'atteindra jamais la 2ième règle qui l'autorise. Gardez bien toujours en mémoire que l'ordre compte. Récupérez l'adresse IP de votre voisin puis exécutez :

root# iptables -I INPUT 1 -p tcp --dport 22 -s \
adresse_ i p_du_voi si n -j ACCEPT

Le '-I INPUT 1' signifie insère en première position dans la liste , l'option -s permet de choisir la source des paquets, on choisit donc les paquets provenant de la machine voisine (on peut également spécifier des réseaux entiers au format 193.54.241.0/24 par exemple).
Essayez de vous connecter sur votre machine depuis la machine voisine, cela devrait marcher. Essayez de vous connecter sur votre machine depuis une autre machine, cela ne devrait pas marcher.
N'hésitez pas à utiliser la commande 'iptables -L -v' pour voir les compteurs de paquets qui évoluent en fonction de vos tests (Nota : vous pouvez utiliser la commande 'iptables -Z' pour remettre à zéro les compteurs de paquets et d'octets).
II existe de très nombreuses options pour construire les règles de filtrage, parmi les plus utilisés on trouve :
-s : sélection de l'adresse IP source (ou réseau source) 'd'ou vient Ie paquet '
-d : sélection de l'adresse IP de destination (ou réseau de destination) 'ou va Ie paquet '
--dport : port destination (Ie port sur lequel le client essaye de se connecter) 'vers quel port a été émis Ie paquet'
--sport: port source (Ie port utilise par Ie client pour créer la connexion) 'depuis quel port a été émis Ie paquet'
-p suivi d'un nom de protocole (cf. Ie fichier /etc/protocols pour la liste complète)
II est également possible d'utiliser Ie symbole ! pour signifier Ie contraire d'une option, par exemple pour dire 'tous les protocoles sauf tcp' : '-p !tcp'.
Nous avons vu comment ajouter et insérer des règles, on peut aussi les effacer en utilisant l'option *-D,* par exemple en faisant :

root# iptables -D INPUT 1

Vous effacez la première règle de la chaîne INPUT. II peut être pratique de voir les numéros de chaque règle lorsqu'on liste les règles, cela se fait avec l'option *-line-numbers :*

iptables -L - v --line-numbers

IPTables a également la possibilité d'avoir des fonctionnalités supplémentaires via un système de modules. Pour spécifier que l'on veut utiliser un module particulier, on utilise l'option '-m' . On trouve par exemple un module « multiport » qui permet de spécifier plusieurs ports sources et / ou destinations dans une seule règle suivant la syntaxe :

```
iptables -A chaine -m multiport -p tcp --dports \
port l ,port2,port3 – j  …..
```

Notez Ie 's' à dports cette fois

Vous trouverez une description des modules existant dans Ie man d'iptables dans la section MATCH EXTENSIONS ou EXTENSIONS DE CORRESPONDANCES (On peut trouver des modules sur internet spécialisés dans Ie filtrage des applications peer-to-peer, des messageries instantanées etc.).
Commencez par effacer toutes les anciennes règles de la chaîne INPUT :

```
# iptables -F  INPUT
```

(plus simplement on pourrait faire 'iptables -F' pour effacer toutes les chaînes de la table filter).
Nous allons maintenant voir comment s'organiser lorsqu'on écrit ses règles de filtrage pour ne pas avoir un 'paté' de règles incompréhensibles par la suite . Pour cela, iptables met à notre disposition la possibilité de rajouter des chaînes utilisateurs. L'idée est donc de pouvoir regrouper nos règles de filtrage en sous ensembles. Par exemple, on peut définir une série de filtrage pour les services web, une autre série pour les accès ssh, une autre pour les accès aux services de messagerie etc .
Pour créer une chaîne utilisateur, il suffit de faire :

```
# iptables -N ma_chaine
```

Depuis la chaîne INPUT, on peut maintenant utiliser l'option -j ma_chaine pour indiquer que l'on souhaite transférer l'étude de ce paquet à la chaîne que l'on a créé,
Imaginons par exemple que l'on veuille gérer plusieurs règles pour contrôler l'accès au service ssh. On pourrait procéder de la manière suivante :
1 - Création d'une chaîne spécifique

```
# iptables -N ssh
```

2 - On indique à la chaîne INPUT que tout ce qui concerne ssh doit être transmis à cette nouvelle chaîne

```
# iptables -A INPUT -p tcp - -dport 22 -j ssh
```

3 - On rajoute nos règles de filtrages dans la chaîne ssh
- on accepte que Ie voisin se connecte

```
# iptables -A ssh -s ip_machine_du_voisin - j ACCEPT
```

- on interdit au reste du monde de se connecter

```
# iptables -A ssh -j DROP
```

Nous avons reproduit Ie même schéma que précédemment mais cette fois nous n'avons mis qu'une seule règle dans la chaîne INPUT. L'idée est de s'organiser pour que l'ensemble des filtres restent compréhensibles 6 mois plus tard quand vous voudrez faire une nouvelle modification et généralement il vaut mieux éviter de trop remplir la chaîne INPUT).

Enfin, il existe 2 options qu'il est bon de connaître :
1) -j LOG --log-prefix 'ce_que_vous_voulez' : cette option déclenchera un message qui sera enregistrer par Ie système, vous Ie retrouverez généralement dans Ie fichier /var/log/messages.

2) -j RETURN permet de sortir de la chaîne courante et de retourner à la chaîne de niveau supérieur pour continuer à travers les règles de celle-ci

a. Exercice 1

A partir de toutes ces informations, essayez d'écrire une liste de règles pour autoriser votre voisin à se connecter sur les ports 22 (ssh) et 25 (srntp, mail) de votre machine. Vous devrez également logguer les connexions ssh que votre machine refusera .

b. Exercice 2 :

Filtrez les paquets sortants de votre machine : interdisez tout traffic web sortant (port 80 et port 443). Vous ne devriez donc plus être capable d'ouvrir une page web avec votre navigateur).
En vous basant ensuite sur la page de man de iptables, trouvez un moyen d'autoriser uniquement I'utilisateur root a faire des connexions sur Ie web (donc root doit pouvoir surfer mais pas un utilisateur standard).

c. Exercice 3 :

Une attaque commune contre un système est Ie ping flood ou différents dénis de service basés sur des paquets qui utilisent Ie protocole ICMP. Trouvez un moyen de limiter Ie nombre de paquets ICMP que votre machine va accepter par secondes. La règle que vous trouverez pourra être utilisée 'sur Ie terrain' au niveau d'une passerelle pour protéger par exemple tout un réseau contre ce type d'attaques.

3) Suivi de connexion (conntrack)

Habituellement, netfilter travaille paquet par paquet, sans se soucier des paquets précédents ou des paquets qui vont suivre. II n'a donc pas de notion de 'session' ou de suivi d'un flux dans Ie temps . Pour fournir cette fonctionnalité, Ie module 'state' a été créé. Le but est de pouvoir identifier les nouveaux flux, les flux établis et les flux étant liés a un autre flux. Ce suivi de connexion permet d'affiner Ie filtrage de certains protocoles.
Le module 'state' définit plusieurs états possibles pour les flux réseaux :
- NEW : c'est une nouvelle connexion
- ESTABLISHED : on connaît déjà cette connexion (elle est passée par l'état NEW il y a peu de temps)
- RELATED : plus subtil, cela permet d'identifier une connexion qui serait liée ou dépendant d'une connexion déjà ESTABLISHED.
- INVALID : tout ce qui n'est pas correctement identifiable (paquets corrompus etc .)

Un exemple concret: le protocole FTP en mode actif.

Par définition, Ie protocole FTP utilise 2 connexions TCP pour communiquer. L'une sert a envoyer les commandes du client vers Ie serveur et I'autre connexion sert à envoyer les données du serveur vers Ie client. Dans son fonctionnement standard, un client FTP se connecte sur Ie port 21 du serveur FTP. Le client indique ensuite au serveur sur quel port il (Ie client) recevra les données . Le serveur FTP va alors établir une connexion depuis son port 20 vers Ie port (>1023) indique par Ie client :

Client FTP (port source aléatoire 'a') ----->Serveur FTP (port 21)
Client FTP(port 'a') ---->Serveur FTP (port 21)
(Ie client indique au serveur FTP sur quel port il devra envoyer les données, par ex 3084)
Serveur FTP(port 20) -----> Client FTP (port 3084)
(Ie serveur transfert des donnees vers Ie client)

Si Ie client est protégé par un firewall netfilter, I'administrateur devrait donc autoriser toutes les connexions venant de l'extérieur vers tous les ports supérieurs a 1023 du client. On perdrait tout l'intérêt du pare-feu avec une telle règle de filtrage vu qu'on 'ouvre' les 3/4 des ports de la machine sur internet. C'est dans ce genre de cas que Ie conntrack est très utile. II va permettre de détecter qu'une connexion FTP est établie (au moment ou Ie client se connecte sur Ie port 21 du serveur) et ouvrir de manière dynamique Ie filtrage pour cette connexion spécifique vers Ie port que Ie client aura choisi.

Démonstration pratique:

Effacez toutes les règles précédentes :
iptables -F

On charge ensuite Ie module noyau qui permet de gérer Ie suivi de connexion FTP:
modprobe ip_conntrack_ftp

On commence par tout interdire :
iptables -A INPUT -j DROP

Testez que les connexions FTP depuis votre machine vers Ie serveur ftp de la salle en mode FTP actif ne fonctionnent pas:
[root@fedora *I]*# ftp
ftp> passive
Passive mode off.
ftp> open **192.168.48.1**
Ca ne marche pas et c'est normal car on DROP tous les paquets entrants pour Ie moment.

Nota : ici on utilise l'adresse IP pour se connecter, la raison est simple, comme on rejette tous les paquets en entrée de notre machine, Ie service de résolution de noms ne fonctionne plus (vous pouvez essayer de trouver la règle qui permettrait de Ie refaire fonctionner, c'est assez simple).

Acceptons maintenant les paquets provenant d'un serveur FTP dans Ie cas d'une connexion que l'on connaît (car c'est nous qui l'avons établie avec notre client FTP) :

```
# iptables -I INPUT 1 -p tcp -- sport 21 -m state - - s t a t e \
ESTABLISHED -j ACCEPT
[root@fedora I]# ftp
ftp> passive
Passive mode off .
ftp> open 193.51 .24.2
login : anonymous, password: rien
```

Cette fois on peut se connecter. La connexion de 'commandes ' sur Ie port 21 est bien fonctionnelle. Une fois connecté, essayez de faire 'Is'. La connexion se bloque En effet, la commande 'Is' en FTP utilise la connexion de 'données' pour transmettre son résultat. Hors pour Ie moment, notre firewall bloque ces connexions.