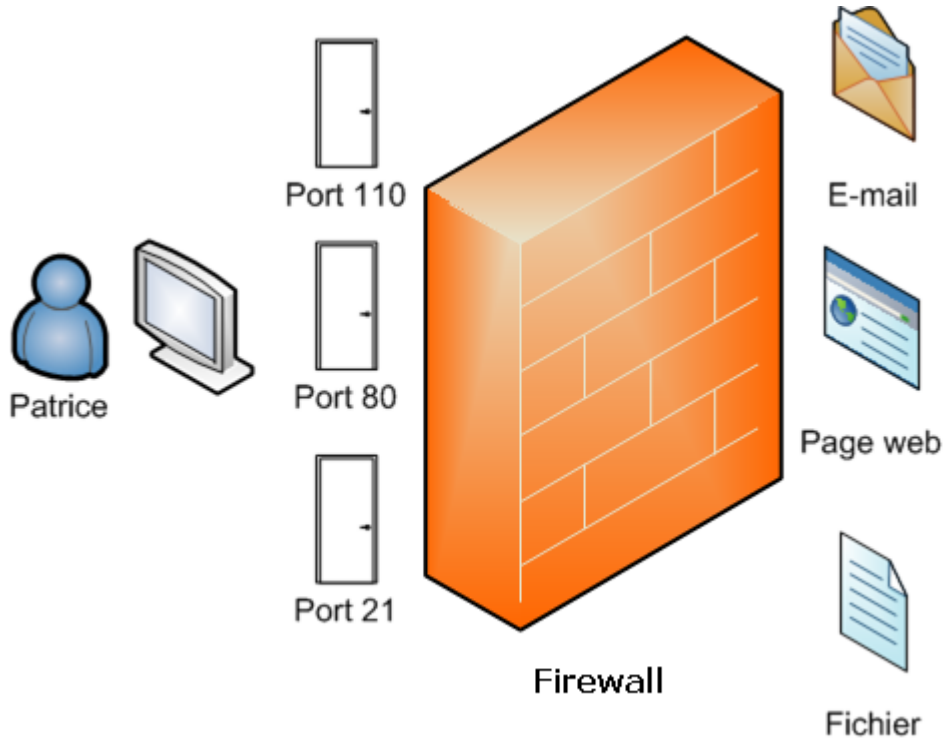
Maintenant que nous savons analyser le trafic réseau et ainsi voir un peu ce qui se passe, nous allons nous atteler au filtrage du trafic à l'aide d'un pare-feu.

Le plus célèbre pare-feu utilisé sous Linux est `iptables`. Il permet d'établir un certain nombre de **règles** pour dire par quels ports *on peut* se connecter à votre ordinateur, mais aussi à quels ports vous *avez le droit* de vous connecter (figure suivante). On peut également filtrer par IP, mais nous ne détaillerons pas cela ici.



Par exemple, si je veux empêcher toute connexion FTP (parce que je considère que le FTP n'est pas sûr), je peux souhaiter bloquer le port 21 (utilisé par FTP).
En général la technique ne consiste pas à bloquer certains ports mais plutôt à bloquer par défaut **tous** les ports et à en autoriser seulement quelques-uns.

Attends… c'est quoi le but, exactement ? Bloquer tout le trafic réseau ? Pour quoi faire ?

C'est avant tout une question de sécurité. Le but d'un pare-feu est d'empêcher que des programmes puissent communiquer sur le réseau sans votre accord. Aujourd'hui, même sous Windows (depuis Windows XP SP2), un pare-feu est intégré par défaut, tant le problème est important.

Avoir un pare-feu ne vous prémunit pas contre les virus (bien que sous Linux, ils restent rares). En revanche, cela rend la tâche **particulièrement difficile** aux pirates qui voudraient accéder à votre machine.

Vous vous souvenez de ce que je vous ai expliqué un peu plus tôt ? Chaque ordinateur possède plusieurs portes d'entrée possibles.
Notre objectif est de bloquer par défaut toutes ces portes et d'autoriser seulement celles dont vous avez besoin, que vous considérez comme « sûres » et que vous utilisez. Par exemple, le port 80 utilisé pour le web est un port sûr que vous pouvez activer.

Notez, et c'est important, qu'il y a des portes d'entrée et des portes de sortie sur votre ordinateur (ce ne sont pas nécessairement les mêmes).

`iptables` est un programme extrêmement puissant, mais tout aussi complexe. Nous ne verrons que des fonctionnalités basiques (et ce sera déjà pas mal). Sachez qu'il peut faire bien plus que ce que l'on va voir : pour en savoir plus, comme d'habitude, lisez le manuel.

`iptables` s'utilise en « root »

Pour manipuler `iptables`, vous devez impérativement être en « root ». Pour la suite des opérations, je vous recommande donc de passer en superutilisateur dès à présent :

console

```
$ sudo su
```

`iptables -L` : afficher les règles

Avec `iptables -L` (attention, un « L » majuscule), vous pouvez afficher les règles qui régissent actuellement le pare-feu :

console

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

On repère trois sections :

- `Chain INPUT` : correspond aux règles manipulant le trafic entrant ;
- `Chain FORWARD` : correspond aux règles manipulant la redirection du trafic ;
- `Chain OUTPUT` : correspond aux règles manipulant le trafic sortant.

Nous ne verrons pas ici la section `FORWARD`. `iptables` permet de rediriger le trafic, mais c'est assez compliqué et ne nous intéresse pas ici. Nous aurons déjà suffisamment de quoi faire avec `INPUT` et `OUTPUT`.

Actuellement, chez moi, les règles sont vides. Il y a trois tableaux mais qui ne contiennent aucune ligne. Par ailleurs, vous noterez à chaque fois les mots `(policy ACCEPT)` qui signifient que, par défaut, tout le trafic est accepté. Donc chez moi, pour le moment, le pare-feu est tout simplement inactif car il ne bloque rien ; mon ordinateur est une vraie passoire. :-D

Si vous avez déjà des règles inscrites dans votre pare-feu (ce qui ne devrait pas être votre cas, mais on ne sait jamais), sachez que vous pouvez les réinitialiser.
Ne le faites que **si vous êtes certains de vouloir le faire**. En effet, sur un ordinateur partagé, peut-être quelqu'un a-t-il déjà configuré le pare-feu et il serait dommage de saboter tout son travail.

console

```
# iptables -F   <-- Attention ! Réinitialise toutes les règles iptables !
```

Le principe des règles

Voici ce que cela pourrait donner lorsqu'on aura établi des règles, par exemple ici pour la section `INPUT` :

```
# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:www
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ssh
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:imap2
```

Première chose à savoir : **l'ordre des règles est important**. En effet, `iptables` les lit de haut en bas et la position de ces règles influe sur le résultat final. Sachez donc que les règles sont numérotées.

Pour avoir les numéros, ajoutez `--line-numbers` :

console

```
# iptables -L --line-numbers
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:www
2    ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ssh
3    ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:imap2
```

Ainsi, la règle filtrant SSH est la règle n° 2.

Chaque ligne correspond à une règle différente qui permet de filtrer ou non une IP ou un port. Parmi les colonnes intéressantes, on note :

- `target` : ce que fait la règle. Ici c'est `ACCEPT`, c'est-à-dire que cette ligne autorise un port et / ou une IP ;
- `prot` : le protocole utilisé (`tcp`, `udp`, `icmp`). Je rappelle que TCP est celui auquel on a le plus recourt. ICMP permet à votre ordinateur de répondre aux requêtes de type « ping » ;
- `source` : l'IP de source. Pour `INPUT`, la source est l'ordinateur distant qui se connecte à vous ;
- `destination` : l'IP de destination. Pour `OUTPUT`, c'est l'ordinateur auquel on se connecte ;
- *la dernière colonne* : elle indique le port après les deux points « : ». Ce port est affiché en toutes lettres, mais avec `-n` vous pouvez obtenir le numéro correspondant.

Sur mon exemple, seuls les ports web, `ssh` et `imap2` (e-mail) sont autorisés en entrée. Personne ne peut se connecter à la machine par un autre biais.
En effet, si vous regardez bien, par défaut j'ai configuré le pare-feu pour qu'il ignore tous les autres paquets : `(policy DROP)`.

Nous allons maintenant apprendre à faire tout cela.

Ajouter et supprimer des règles

Voici les principales commandes à connaître.

- `-A chain` : ajoute une règle en fin de liste pour la `chain` indiquée (`INPUT` ou `OUTPUT`, par exemple).
- `-D chain rulenum` : supprime la règle n° `rulenum` pour la `chain` indiquée.
- `-I chain rulenum` : insère une règle au milieu de la liste à la position indiquée par `rulenum`. Si vous n'indiquez pas de position `rulenum`, la règle sera insérée en premier, tout en haut dans la liste.
- `-R chain rulenum` : remplace la règle n° `rulenum` dans la `chain` indiquée.

- `-L` : liste les règles (nous l'avons déjà vu).
- `-F chain` : vide toutes les règles de la `chain` indiquée. Cela revient à supprimer toutes les règles une par une pour cette `chain`.
- `-P chain regle` : modifie la règle par défaut pour la `chain`. Cela permet de dire, par exemple, que par défaut tous les ports sont fermés, sauf ceux que l'on a indiqués dans les règles.

De manière générale, l'ajout d'une règle se passe suivant ce schéma :

console
```
iptables -A (chain) -p (protocole) --dport (port) -j (décision)
```

Remplacez `chain` par la section qui vous intéresse (`INPUT` ou `OUTPUT`), `protocole` par le nom du protocole à filtrer (TCP, UDP, ICMP…) et enfin `décision` par la décision à prendre : `ACCEPT` pour accepter le paquet, `REJECT` pour le rejeter ou bien `DROP` pour l'ignorer complètement.

Le mieux est de découvrir comment on ajoute une règle par une série d'exemples. 

console
```
# iptables -A INPUT -p tcp --dport ssh -j ACCEPT
```

Cela ajoute à la section `INPUT` (donc, pour le trafic entrant) une règle sur les données reçues via le protocole TCP sur le port de `ssh` (vous pouvez remplacer `ssh` par le numéro du port, soit 22). Lorsque votre ordinateur recevra des données en TCP sur le port de SSH, celles-ci seront acceptées ; cela vous permettra donc de vous connecter à distance à votre PC via SSH.

Vous pouvez faire de même avec d'autres ports :

console
```
# iptables -A INPUT -p tcp --dport www -j ACCEPT
```

… pour le web (80).

console
```
# iptables -A INPUT -p tcp --dport imap2 -j ACCEPT
```

… pour les mails, etc.

 Si vous ne précisez pas de port (en omettant la section `dport`), tous les ports seront acceptés !

Autoriser les pings

En plus d'autoriser le trafic sur ces ports, je peux vous conseiller d'autoriser le protocole ICMP (pour pouvoir faire un ping) sur tous ces derniers :

console
```
# iptables -A INPUT -p icmp -j ACCEPT
```

Comme je n'ai pas indiqué de section `--dport`, cette règle s'applique à tous les ports, **mais pour les pings (icmp) uniquement** !

Votre ordinateur répondra alors aux « pings » pour indiquer qu'il est bien en vie.

Vos règles `iptables` pour `INPUT` devraient maintenant ressembler à ceci :

console
```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
```

```
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:www
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ssh
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:imap2
ACCEPT     icmp --  anywhere             anywhere
```

Autoriser les connexions locales et déjà ouvertes

Pour l'instant, nos règles sont encore un peu trop restrictives et pas vraiment utilisables (vous risquez de ne plus pouvoir faire grand-chose).
Je vous propose d'ajouter deux règles pour « assouplir » un peu votre pare-feu et le rendre enfin utilisable.

console

```
# iptables -A INPUT -i lo -j ACCEPT
# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Ces deux règles utilisent des options un peu différentes de celles que nous avons vues jusqu'ici. Voici quelques explications.

1. La première règle autorise tout le trafic sur l'interface de loopback locale grâce à `-i lo`. Il n'y a pas de risque à autoriser votre ordinateur à communiquer avec lui-même, d'autant plus qu'il en a parfois besoin !

2. La seconde règle autorise toutes les connexions qui sont déjà à l'état `ESTABLISHED` ou `RELATED`. En clair, elle autorise toutes les connexions qui ont été demandées par votre PC. Là encore, cela permet d'assouplir le pare-feu et de le rendre fonctionnel pour une utilisation quotidienne.

Refuser toutes les autres connexions par défaut

Il reste un point essentiel à traiter car, **pour l'instant, ce filtrage ne sert à rien**. En effet, nous avons indiqué quelles données nous autorisions, mais **nous n'avons pas dit que toutes les autres devaient être refusées** !

Changez donc la règle par défaut pour `DROP` par exemple :

console

```
# iptables -P INPUT DROP
```

`iptables` devrait maintenant indiquer que par défaut tout est refusé, sauf ce qui est indiqué par les lignes dans le tableau :

console

```
# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:www
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ssh
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:imap2
ACCEPT     icmp --  anywhere             anywhere
```

Le filtrage est radical. Nous n'avons pas autorisé beaucoup de ports et il se pourrait que vous vous rendiez compte que certaines applications n'arrivent plus à accéder à l'internet (normal, leur port doit être filtré).

À vous de savoir quels ports ces applications utilisent pour modifier les règles en conséquence.
Au besoin, pensez à faire de même pour les règles de sortie (`OUTPUT`).

Appliquer les règles au démarrage

Si vous redémarrez votre ordinateur, les règles `iptables` auront disparu !
Le seul moyen pour qu'elles soient chargées au démarrage consiste à créer un script qui sera exécuté au

démarrage.

Justement, ça tombe bien, nous allons étudier la programmation de scripts shell sous Linux dans la prochaine partie. :-)
En attendant, si vous voulez lire un mode d'emploi rapide pour mettre les règles au démarrage, je vous invite à lire la documentation `ubuntu-fr`.

Comme vous avez pu le constater, `iptables` est donc un pare-feu assez compliqué. Sachez que des développeurs ont travaillé sur un programme qui simplifie l'utilisation d'`iptables` : **ufw** (*Uncomplicated Firewall*). Contrairement à `iptables`, ce programme n'est pas disponible partout, mais on le trouve dans les versions récentes d'Ubuntu.

En résumé

- Sur l'internet, chaque ordinateur est identifié par une adresse IP. Par exemple : `86.172.120.28`.
- On peut associer à chaque adresse IP un nom d'hôte, plus facile à retenir, comme `lisa.simple-it.fr`. Écrire le nom d'hôte est équivalent à écrire l'adresse IP.
- La commande `host` permet de traduire une IP en nom d'hôte et inversement.
- `ifconfig` liste les interfaces réseau (cartes réseau) de votre machine et permet de les configurer ainsi que de les activer.
- `netstat` affiche la liste des connexions ouvertes sur votre machine. Elle indique notamment quel port est utilisé à chaque fois, le port représentant en quelque sorte la porte d'entrée à votre machine.
- Il est possible de bloquer l'accès à certains ports avec le programme `iptables`, un pare-feu (*firewall*) très puissant. Celui-ci est cependant assez complexe à configurer.