Reprenez le contrôle à l'aide de Linux !

30 heures Facile

Mis à jour le 29/06/2021

Analyser le réseau et filtrer le trafic avec un pare-feu

Ce chapitre vous propose d'apprendre à maîtriser le trafic réseau qui passe par votre ordinateur. En effet, lorsque vous êtes connectés à l'internet, vous avez régulièrement des applications qui vont se connecter puis télécharger et envoyer des informations. Comment surveiller ce qui se passe ? Quelle application est en train de communiquer et sur quel port ?

Savoir paramétrer un pare-feu est essentiel, que ce soit sur votre PC à la maison ou, à plus forte raison, sur un serveur. Cela vous protège de manière efficace contre les programmes qui voudraient échanger des informations sur le réseau sans votre accord. C'est une mesure de sécurité essentielle qu'il faut connaître et dont aucun administrateur système sérieux ne peut se passer.

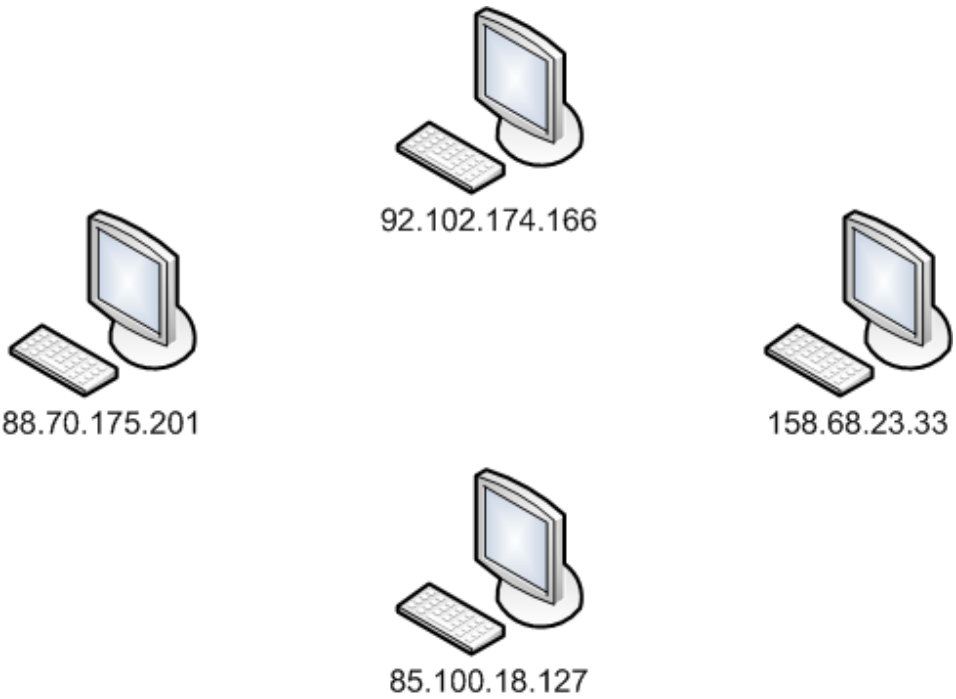
Je vous propose de découvrir d'abord quelques outils de base qui vont vous permettre de bien comprendre comment une IP est associée à un nom d'hôte. Puis nous analyserons le trafic en cours avec un outil comme `netstat`. Enfin — et ce ne sera pas le plus facile, je vous préviens — nous apprivoiserons le célèbre pare-feu utilisé sous Linux : `iptables`. Il est assez complexe à paramétrer, mais heureusement des programmes supplémentaires peuvent nous simplifier le travail.

host & whois : qui êtes-vous ?

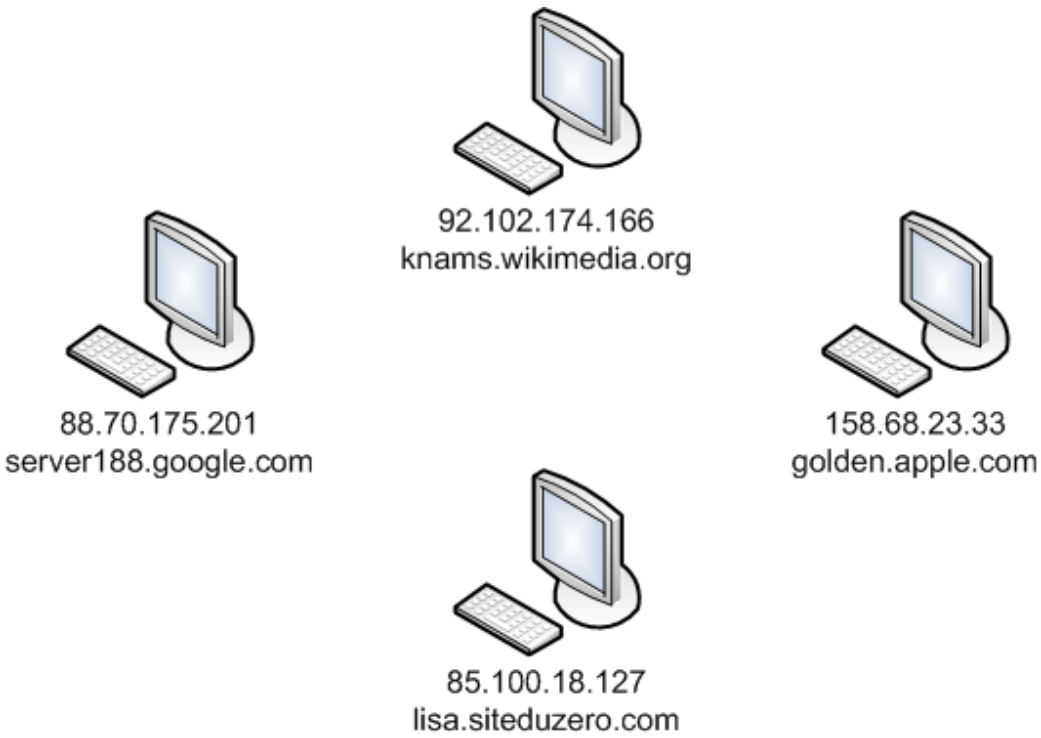
Comme vous le savez sûrement, chaque ordinateur relié à l'internet est identifié par une adresse IP (figure suivante).

Une adresse IP est une suite de quatre nombres séparés par des points. Par exemple : `86.172.120.28`.

Cette adresse est au format **IPv4**. À l'heure actuelle, c'est encore le type d'IP le plus utilisé, mais ces adresses sont appelées petit à petit à être remplacées par la norme **IPv6**. Bientôt, tout le monde aura donc une IP qui ressemblera plutôt à quelque chose comme ceci : `fe80::209:62fa:fb80:29f2` .



On peut associer à chaque IP ce qu'on appelle un nom d'hôte (`hostname`). C'est un nom en toutes lettres plus facile à mémoriser et qui revient exactement au même que d'écrire l'adresse IP, comme le suggère la figure suivante.



Chaque serveur peut ainsi avoir un nom d'hôte plus facile à retenir. Je retiens mieux le nom de notre serveur (`lisa.simple-it.fr`) que son équivalent en adresse IP. :-)

Convertir une IP en nom d'hôte et inversement

Il existe une commande qui est capable d'effectuer la conversion dans les deux sens :

- à partir d'une IP on peut avoir le nom d'hôte correspondant ;
- à partir d'un nom d'hôte, on peut avoir l'IP correspondante.

Cette commande, c'est `host` . Donnez-lui en paramètre une IP ou un nom d'hôte.

Par exemple :

```
$ host siteduzero.com
siteduzero.com has address 92.243.25.239
siteduzero.com mail is handled by 0 mail.siteduzero.com
```

La commande nous répond que l'IP de `siteduzero.com` est `92.243.25.239` . Elle nous indique par ailleurs le nom du serveur qui gère les e-mails.

Maintenant, essayons à l'envers avec l'IP :

```
$ host 92.243.25.239
123.219.248.80.in-addr.arpa domain name pointer lisa.simple-it.fr.
```

On nous répond que le nom d'hôte de `92.243.25.239` est `lisa.simple-it.fr` .

Mais, je croyais que c'était `siteduzero.com` cette IP ?

Oui, en fait il s'agit d'un synonyme dans le cas présent :\\ `siteduzero.com = lisa.simple-it.fr` .

Vous pouvez essayer la même manipulation avec d'autres IP et noms d'hôte : prenez des sites que vous connaissez comme par exemple `mozilla.org` , `google.fr` , etc.

Gérer les noms d'hôte personnalisés

Les associations entre les IP et les noms d'hôte sont faites sur ce que l'on appelle des *serveurs DNS*. Nous n'allons pas entrer dans le détail, mais sachez en gros que chaque fournisseur d'accès met en place des serveurs DNS qui fournissent la liste des correspondances IP ↔ noms d'hôte.
Si vous voulez en découvrir plus sur le fonctionnement des DNS, je vous invite à lire mon tutoriel sur les DNS.

Ainsi, lorsque vous tapez `siteduzero.com` dans votre navigateur, vous pouvez obtenir l'adresse IP correspondante et naviguer sur le Site du Zéro.
C'est quand même plus pratique que d'avoir à retenir l'IP !

Vous ne pouvez pas modifier la liste des correspondances IP ↔ noms d'hôte sur le serveur DNS (puisque ce serveur est utilisé par de nombreuses personnes), mais en revanche vous pouvez établir une liste de correspondances personnalisée sur votre ordinateur.

Ouvrez pour cela en root le fichier `/etc/hosts` :

```
$ sudo nano /etc/hosts
```

Dedans, vous devriez avoir des lignes ressemblant à ceci :

```
127.0.0.1       localhost
127.0.1.1       mateo21-laptop
```

À gauche l'IP, à droite le nom d'hôte correspondant. Écrire `localhost` est donc équivalent à écrire `127.0.0.1` .

Vous pouvez ajouter des lignes sur le même modèle pour faire correspondre une IP à un nom d'hôte.

Quel intérêt ? Cela dépend. Parfois, les DNS ne fonctionnent pas bien pendant de courtes périodes (c'est très rare, mais ça peut arriver). Dans ce cas, il est plus simple de modifier votre fichier `hosts` pour pouvoir continuer à consulter votre site préféré en « forçant » l'association du nom d'hôte et de l'IP.

Vous pourriez donc ajouter :

```
92.243.25.239      siteduzero.com
```

Enregistrez, ouvrez un navigateur, puis tapez `siteduzero.com` pour voir si ça fonctionne.

Attention : cette technique a l'avantage de forcer l'association, mais si notre serveur change un jour d'IP, votre ordinateur ne sera pas au courant ! En règle générale, il est préférable d'utiliser les serveurs DNS qui se mettent régulièrement à jour (une fois par jour, en moyenne) afin d'avoir toujours une liste actualisée.

Sur un réseau local, il peut être pratique d'associer un nom d'hôte à chaque PC pour pouvoir vous y connecter sans avoir à retenir l'IP :

```
192.168.0.5      pc-papa
```

Ainsi, écrire `pc-papa` vous permet d'accéder à cet ordinateur sans avoir à retenir l'adresse IP correspondante.

`whois` : tout savoir sur un nom de domaine

Chaque nom de domaine doit obligatoirement indiquer qui se trouve derrière : nom, prénom, adresse et moyens de contact. C'est une règle.

L'outil `whois` vous permet d'obtenir facilement ces informations pour n'importe quel nom de domaine :

```
$ whois siteduzero.com

[...]

domain: siteduzero.com
reg_created: 2002-06-09 21:53:29
expires: 2011-06-09 21:53:29
created: 2007-02-27 06:56:43
changed: 2010-04-13 15:35:32
transfer-prohibited: yes
ns0: a.dns.gandi.net
ns1: b.dns.gandi.net
ns2: c.dns.gandi.net
owner-c:
  nic-hdl: PD2500-GANDI
  owner-name: Simple IT SARL
  organisation: Simple IT SARL
  person: Pierre DUBUC
  address: 23 Rue Le Peletier
  zipcode: 75009
  city: Paris
  country: France
lastupdated: 2010-05-17 10:27:41

[...]
```

Utilisez ces informations avec parcimonie. En général, on y a recours lorsque l'on a besoin de contacter le propriétaire d'un nom de domaine ou d'une adresse IP, pour régler un litige mettant en jeu le nom de domaine ou l'IP en question par exemple.

ifconfig & netstat : gérer et analyser le trafic réseau

Nous allons découvrir ici deux commandes : `ifconfig` et `netstat` . La première permet de gérer les connexions réseau de votre machine (pour les activer / désactiver, par exemple) tandis que la

seconde vous permet d'analyser ces connexions, de connaître des statistiques, etc.

`ifconfig` : liste des interfaces réseau

Votre ordinateur possède en général plusieurs **interfaces réseau**, c'est-à-dire plusieurs moyens de se connecter au réseau.

Tapez `ifconfig` dans la console pour voir ce que ça donne :

```
$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:90:f5:56:44:5a
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          Packets reçus:0 erreurs:0 :0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          Octets reçus:0 (0.0 B) Octets transmis:0 (0.0 B)
          Interruption:220 Adresse de base:0xe000

lo        Link encap:Boucle locale
          inet adr:127.0.0.1  Masque:255.0.0.0
          adr inet6: ::1/128 Scope:Hôte
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          Packets reçus:10 erreurs:0 :0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:0
          Octets reçus:500 (500.0 B) Octets transmis:500 (500.0 B)

wlan0     Link encap:Ethernet  HWaddr 00:19:d2:61:90:0a
          inet adr:192.168.1.2  Bcast:192.168.1.255  Masque:255.255.255.0
          adr inet6: fe80::219:d2ff:fe61:900a/64 Scope:Lien
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Packets reçus:5238 erreurs:0 :0 overruns:0 frame:0
          TX packets:4899 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          Octets reçus:5069449 (5.0 MB) Octets transmis:1202459 (1.2 MB)
```

On distingue ici trois interfaces réseau. Vous en avez peut-être plus, peut-être moins ; tout dépend de votre ordinateur.

Les interfaces que j'ai sont assez courantes, détaillons-les :

- `eth0` : cela correspond à la connexion par câble réseau (ce qu'on appelle en général le *câble RJ45* – figure suivante). Si votre PC est relié au réseau via un câble, c'est sûrement ce moyen de communication que vous utilisez actuellement. Notez que certains ordinateurs (et notamment les serveurs) ont plusieurs sorties réseau filaires. Dans ce cas, vous devriez voir aussi des

interfaces `eth1` , `eth2` , etc.

- `lo` : c'est la boucle locale. Tout le monde devrait avoir cette interface. Elle correspond à une connexion à... vous-mêmes. C'est pour cela qu'on l'appelle la boucle locale : tout ce qui est envoyé par là vous revient automatiquement. Cela peut paraître inutile, mais on a parfois besoin de se connecter à soi-même pour des raisons pratiques.
- `wlan0` : il s'agit d'une connexion sans-fil type Wi-Fi. Là encore, bien que ce soit plus rare, si vous avez plusieurs cartes réseau sans fil, vous aurez un `wlan1` , `wlan2` , etc.

Observez les résultats de ma commande et essayez de deviner par quelle interface réseau je me connecte à l'internet.

...

Vous avez trouvé ? Il ne fallait pas avoir peur de lire le détail des messages.
En effet, bien que je possède une sortie réseau filaire (RJ45), j'utilise ici le Wi-Fi, comme en témoigne la ligne `Packets reçus:5238` pour le Wi-Fi `wlan0` (alors qu'il y en a 0 pour `eth0`). C'est donc l'interface active que j'utilise le plus.

La commande `ifconfig` permet aussi de faire des réglages réseau. Toutefois, cela sortirait un peu du cadre de ce cours et il vous faudrait des connaissances en réseau pour bien l'utiliser.
Voici cependant un réglage très simple que vous pouvez faire et qui vous sera probablement utile : l'activation / désactivation d'interface.

Il suffit d'écrire une commande sous cette forme :

```
ifconfig interface etat
```

Remplacez :

- `interface` par le nom de l'interface que vous voulez modifier (`eth0` , `wlan0` ...) ;
- `etat` par `up` ou `down` selon si vous voulez activer ou désactiver l'interface.

Exemple :

```
$ ifconfig eth0 down
```

... désactive l'interface `eth0` (filaire). Plus aucun trafic ne pourra alors circuler par l'interface `eth0` .

```
$ ifconfig eth0 up
```

... la réactive de nouveau.

Vous aurez peut-être besoin de connaître ces commandes un jour ou l'autre si vous devez désactiver puis réactiver une interface pour prendre en compte des changements dans la configuration de votre réseau.

`netstat` : statistiques sur le réseau

La commande `netstat` risque de vous paraître un peu complexe si vous avez peu de connaissances concernant les réseaux, mais elle est incontournable quand on veut savoir ce que notre machine est en train de faire sur le réseau.

`netstat` peut afficher beaucoup d'informations. Pour sélectionner celles qui nous intéressent, on a recours à de nombreux paramètres.
Plutôt que de les expliquer un par un, je vais vous montrer quelques combinaisons de paramètres qui donnent des résultats intéressants.

`netstat -i` : statistiques des interfaces réseau

Pour commencer, essayez l'option `-i` :

```
$ netstat -i
Table d'interfaces noyau
Iface      MTU Met    RX-OK  RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0       1500 0         0      0      0 0             0      0      0      0  BMU
lo        16436 0        10      0      0 0            10      0      0      0  LRU
wlan0      1500 0      5161      0      0 0          4810      0      0      0  BMRU
```

Vous n'aurez pas nécessairement les mêmes lignes que moi ; tout dépend de votre ordinateur.
Il s'agit là d'un tableau présentant, pour chaque interface réseau que vous avez, une série de statistiques d'utilisation. On retrouve ici nos interfaces `eth0` , `lo` et `wlan0` .

Comme vous le voyez sur la colonne `RX-OK` , c'est `wlan0` qui est l'interface la plus active. Et vous noterez que `lo` est un petit peu utilisée elle aussi ; comme quoi se connecter à soi-même peut s'avérer utile.

Je ne rentrerai pas dans le détail de ces colonnes car c'est assez technique, mais vous savez au moins détecter l'activité de vos interfaces grâce à cette commande.

`netstat -uta` : lister toutes les connexions ouvertes

```
$ netstat -uta
```

```
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp        0      0 mateo21-laptop.lo:60997 debian-mirror.mirro:ftp ESTABLISHE
tcp        1      0 mateo21-laptop.lo:33721 lisa.simple-it.fr:www CLOSE_WAIT
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
udp        0      0 *:bootpc                *:*
udp        0      0 *:mdns                  *:*
udp        0      0 *:45176                 *:*
```

Les options signifient :

- `-u` : afficher les connexions UDP ;
- `-t` : afficher les connexions TCP ;
- `-a` : afficher toutes les connexions quel que soit leur état.

TCP et UDP sont deux protocoles différents pour envoyer des données sur le réseau.
UDP est plutôt utilisé dans les jeux en réseau et pour les communications vocales (avec Skype, par exemple). Sinon, de manière générale, TCP est le protocole le plus utilisé. Je n'irai pas plus loin dans les explications mais vous pouvez vous renseigner si le sujet vous intéresse.

Pour filtrer un peu, on va enlever les connexions UDP qui, la plupart du temps, sont moins importantes :

```
$ netstat -ta
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp        0      0 mateo21-laptop.lo:60997 debian-mirror.mirro:ftp ESTABLISHE
tcp        0   4107 mateo21-laptop.lo:33721 lisa.simple-it.fr:www ESTABLISHED
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
```

Ce tableau vous indique qui, depuis l'adresse locale, est connecté à qui (à une adresse distante).

Chaque connexion a un état. Ici, on repère les états `LISTEN` et `ESTABLISHED` .
De nombreux états sont possibles ; en voici quelques-uns à connaître :

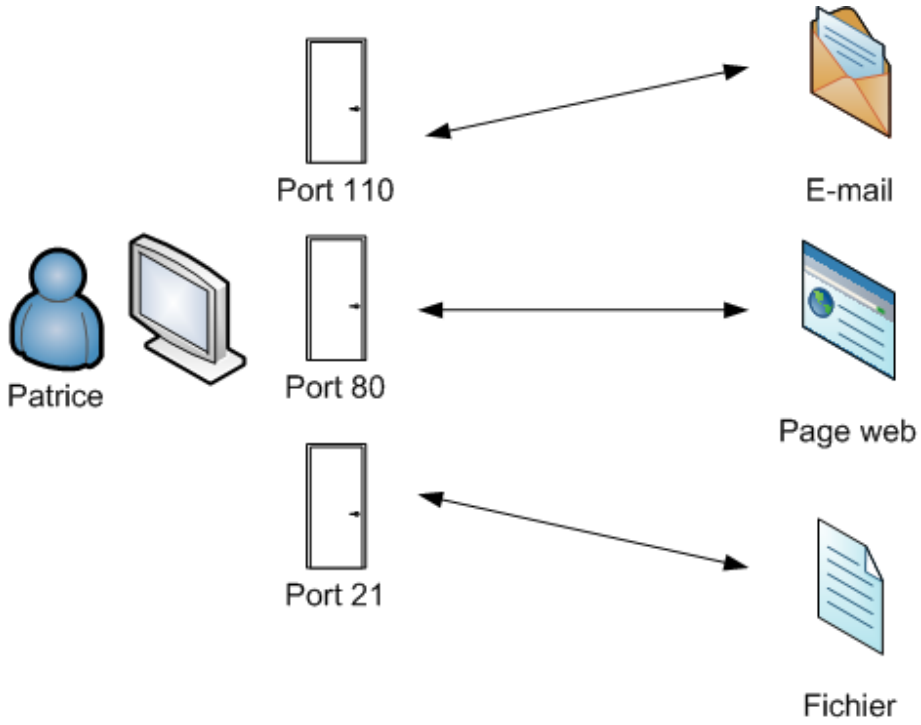
- `ESTABLISHED` : la connexion a été établie avec l'ordinateur distant ;
- `TIME_WAIT` : la connexion attend le traitement de tous les paquets encore sur le réseau avant de commencer la fermeture ;
- `CLOSE_WAIT` : le serveur distant a arrêté la connexion de lui-même (peut-être parce que vous

êtes restés inactifs trop longtemps ?) ;

- `CLOSED` : la connexion n'est pas utilisée ;
- `CLOSING` : la fermeture de la connexion est entamée mais toutes les données n'ont pas encore été envoyées ;
- `LISTEN` : à l'écoute des connexions entrantes.

Il y en a d'autres que vous pouvez lire dans la documentation. Globalement, ce qu'il faut retenir, c'est que les connexions à l'état `LISTEN` ne sont pas utilisées actuellement mais qu'elles « écoutent » le réseau au cas où quelqu'un veuille se connecter à votre ordinateur.

Regardez en particulier le **port** sur lequel ces connexions écoutent (après le symbole « : ») car c'est probablement l'information la plus intéressante. En effet, on peut se connecter à chaque ordinateur via différentes « portes » appelées *ports*. Chaque service utilise un port différent, comme l'illustre la figure suivante.



À la première ligne, vous avez `*:ssh` , ce qui signifie que SSH est en train d'écouter sur le port de SSH au cas où quelqu'un veuille se connecter à votre machine. C'est logique puisque j'ai activé le serveur SSH pour pouvoir m'y connecter à distance au besoin.

D'autres connexions, elles, sont déjà établies et donc en cours d'utilisation. Par exemple, au niveau de l'adresse distante, je suis connecté par FTP à `debian-mirror.mirro:ftp` et je suis connecté à un serveur web `lisa.simple-it.fr:www` .
En clair, je suis en train de charger une page sur le Site du Zéro.

Vous pouvez ajouter `-n` si vous désirez avoir les numéros des ports plutôt qu'une description en

toutes lettres :

```
$ netstat -tan
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp       15      0 192.168.1.2:60997       128.101.240.212:21      CLOSE_WAIT
tcp        0      0 192.168.1.2:54001       80.248.219.123:80       ESTABLISHE
tcp6       0      0 :::22                   :::*                    LISTEN
```

Cela correspond aux ports que l'on connaît : 22 pour SSH, 21 pour FTP, 80 pour le web, etc.

`netstat -lt` : liste des connexions en état d'écoute

Très utile, l'option `-l` vous permet de filtrer les connexions à l'état `LISTEN` et donc de savoir quels ports de serveur sont susceptibles d'être utilisés en ce moment sur votre machine.

```
$ netstat -lt
Connexions Internet actives (seulement serveurs)
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
```

`netstat -s` : statistiques résumées

Enfin, si vous êtes très friands de statistiques réseau, `-s` est fait pour vous :

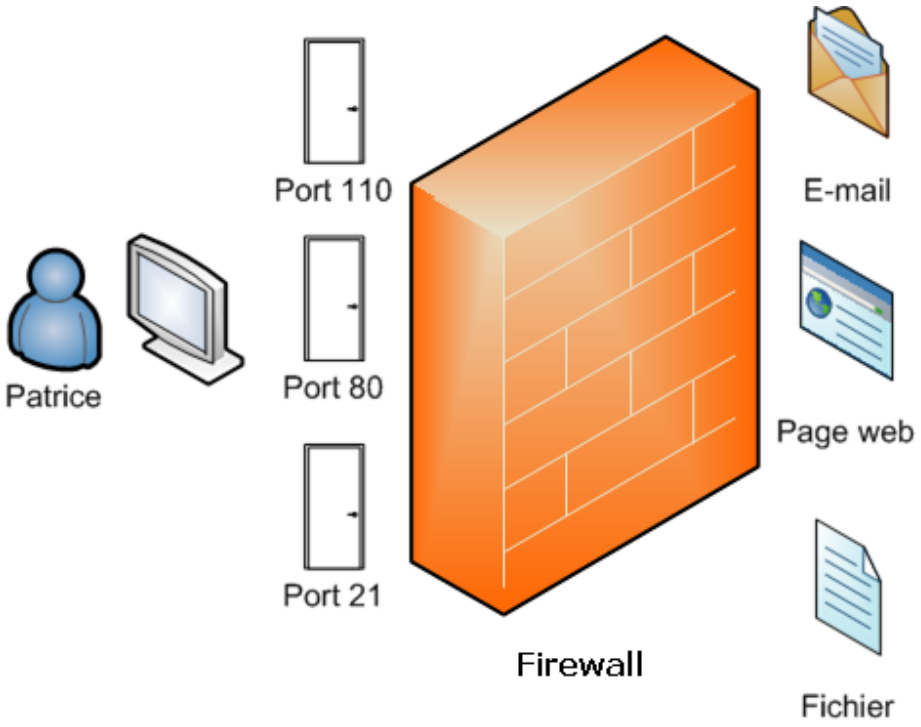
```
$ netstat -s
Ip:
    7443 paquets reçus au total
    1 avec des en-têtes invalides
    8 avec des adresses invalides
    0 réacheminés
    0 paquets arrivant rejetés
    7354 paquets entrants délivrés
    7226 requêtes envoyées
Icmp:
    0 Messages ICMP reçus
    0 messages ICMP entrant échoués

[...]
```

iptables : le pare-feu de référence

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 :

```
$ 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 :

```
# 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.

```
# 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` :

```
# 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 :

```
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.

```
# 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 :

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

... pour le web (80).

```
# 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 :

```
# 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 :

```
# 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.

```
# 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 :

```
# 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 :

```
# 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.

J'AI TERMINÉ CE CHAPITRE ET JE PASSE AU SUIVANT

TRANSFÉRER DES FICHIERS

COMPILER UN PROGRAMME DEPUIS LES SOURCES

Le professeur

Mathieu Nebra

Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms :o)

Découvrez aussi ce cours en...

Livre PDF

FAQ

POUR LES ENTREPRISES

Former et recruter

EN PLUS

Boutique

Mentions légales

Conditions générales d'utilisation

Politique de protection des données personnelles

Cookies

Accessibilité

Français

Télécharger dans l'App Store