Administration Système sous Linux (Ubuntu Server)

Grégory Morel

2022-2023

CPE Lyon - 3IRC / 4ETI / 3ICS

Cours 6

Réseau et sécurité

Rappels : adressage IPv4

Classes d'adresses IPv4

Rappels

Adresse IPv4 = 4 octets séparés par des points : `n1.n2.n3.n4`

2 parties : net id et host id

2 hôtes ayant même *net id* communiquent directement ; sinon il faut un routage

Classe	net id	1ers bits	1ère adresse	Dernière adresse	Nb réseaux	Nb hôtes
A	1 octet	0	0.0.0.0	127.255.255.255	2^7	2^{24}
B	2 octets	10	128.0.0.0	191.255.0.0	2^{14}	2^{16}
C	3 octets	110	192.0.0.0	223.255.255.0	2^{21}	2^8
D [1]	indéf.	1110	224.0.0.0	239.255.255.255	2^{28} adresses	
E [2]	indéf.	1111	240.0.0.0	255.255.255.255	2^{28} adresses	

Obsolète, mais encore utilisé !

1. Adresses de multidiffusion *(multicast)*
2. Réservée pour un usage futur

Adresses privées

Chaque classe comporte une plage d'adresses non routables et réservées aux réseaux locaux / privés :

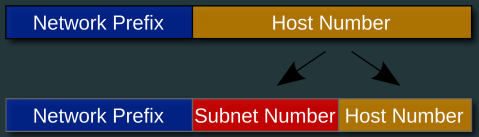
- `10.0.0.0` à `10.255.255.255`
- `172.16.0.0` à `172.31.255.255`
- `192.168.0.0` à `192.168.255.255`
- `127.0.0.1` est l'adresse de bouclage *(loopback)* : elle représente la machine elle-même

Par ailleurs,

- si l'*host id* ne contient que des `0`, l'adresse désigne le réseau lui-même
- si l'*host id* ne contient que des `1`, c'est une adresse de broadcast (permet d'envoyer une trame vers tous les hôtes du réseau)

Sous-réseaux

Niveau hiérarchique intermédiaire entre le réseau et les hôtes



Exemple

Une adresse de la classe B peut être vue comme 256 réseaux de 254 machines, plutôt que comme un seul réseau de 65 534 machines[1]

Pour communiquer entre elles, les machines doivent appartenir à un même réseau ou sous-réseau !

1. Chaque réseau ayant deux adreses réservées (`.0` et `.255`), on gagne en flexibilité d'adressage mais on perd en nombre de machines adressables (ici, 65024 vs 65534)

Masque de sous-réseau

Masque de sous-réseau

Masque binaire permettant de distinguer l'adresse de réseau et de sous-réseau de l'adresse de l'hôte dans une adresse IP :

$$\text{Adresse réseau} = \text{Adresse IP} \;\&^{1}\; \text{Masque de sous-réseau}$$

Calculer un masque de sous-réseau :

1. Déterminer N le nombre de machines dans le réseau, et ajouter 2
2. Trouver le plus petit p tel que $2^p \geq N$
3. Le masque de sous-réseau est constitué de $32 - p$ chiffres *1* suivis de p chiffres *0*

Il existe donc 32 masques de sous-réseau possibles[2]

1. Il s'agit ici du "ET" binaire
2. Autrefois, on évitait les deux masques constitués uniquement de 0 ou de 1

Notation CIDR (Classless Inter-Domain Routing)

Intérêt

- Notion de classes devenue obsolète
- Représentation compacte d'une plage d'adresses IP
- Diminue la taille des tables de routage

Notation : nombre de bits correspondant au sous-réseau dans l'adresse IP, précédés d'un "slash"

Exemple

La notation CIDR /19 fait référence au masque `11111111.11111111.11100000.00000000` soit 255.255.224.0

Exemple

- Quelle est l'adresse de sous-réseau de la machine `91.198.174.2/19` ?

	Notation binaire	Notation décimale
	01011011.11000110.10101110.00000010	91.198.174.2
&	11111111.11111111.11100000.00000000	255.255.224.0
=	01011011.11000110.10100000.00000000	91.198.160.0

- Quelle est l'adresse de l'hôte au sein de ce sous-réseau ?

	Notation binaire	Notation décimale
	01011011.11000110.10101110.00000010	91.198.174.2
&	00000000.00000000.00011111.11111111	255.255.224.0
=	00000000.00000000.00001110.00000010	0.0.14.2

Exemple

- Comment subdiviser un 192.44.78.0/24 en 4 sous-réseaux ? Combien de machines seront adressables sur chaque sous-réseau ?
  ⇒ On a besoin de $\log_2(4) = 2$ bits pour distinguer les sous-réseaux. Les 4 sous-réseaux sont donc :
  `192.44.78.0/26`
  `192.44.78.64/26`
  `192.44.78.128/26`
  `192.44.78.198/26`
  ⇒ Il reste 6 bits pour adresser les machines dans chaque réseau, soit $2^6 - 2 = 62$ adresses possibles
- On veut subdiviser le réseau 192.168.1.0/25 en sous-réseaux de 50 machines. Quel est le masque de sous-réseau ?
  ⇒ $2^5 < 50 + 2 \leq 2^6$ d'où $p = 6$. Le masque de sous-réseau est donc `11111111.11111111.11111111.11000000` soit 255.255.255.192

IPv6

Les adresses IPv4 sont arrivées à saturation le 3 février 2011 :

- mauvaise gestion initiale
- multiplication de la demande des particuliers
- explosion des dispositifs mobiles, des objets connectés...

⇒ une solution possible est le NAT *(Network Address Translation)*, qui permet à des machines d'un sous-réseau privé de communiquer avec le reste d'Internet
⇒ autre solution : passage à IPv6 :

- adresses sur 128 bits (chaque humain peut en posséder des milliards de milliards)
- notation : 8 groupes de 2 octets écrits en hexa et séparés par des : (ex. : `2001:0e36:2ed9:d4f0:021b:(0000):(0000):f81b`)

Encore en transition en 2020

Configuration réseau sous Ubuntu

Nomenclature des interfaces réseau

Classiquement, les interfaces réseau étaient nommées `eth0`, `eth1`… par le noyau

Problème

Les noms pouvaient changer après un redémarrage de la machine ou une modification de matériel (par exemple, `eth0` devient `eth1` et devient autorisée par le pare-feu !)

Une solution est de nommer les interfaces d'*après leur adresse MAC*. Cependant : les adresses MAC ne sont pas toujours fixes (ex. : machines virtuelles)

Solution

Predictable Network Interface Names : le nom est choisi par le BIOS en fonction de l'emplacement sur la carte mère (ex. : `enp0s3`, pour *ethernet network peripheral 0 serial 3*)

Utilitaires de configuration réseau sous Ubuntu

- Jusqu'à Debian 9 : paquets `net-tools` et `wireless-tools` (`ifconfig`, `route`, `arp`, `netstat`…)
- Depuis Debian 9 : dépréciés[1] et remplacés respectivement par `iproute2` et `iw` ; syntaxe des différentes commandes unifiée :

Utilisation	Commande net-tools	Commande iproute2
Adressage	`ifconfig`	`ip addr`, `ip link`
Routage	`route`	`ip route`
Résolution d'adresses	`arp`	`ip neigh`
VLAN	`vconfig`	`ip link`
Tunneling	`iptunnel`	`ip tunnel`
Multicast	`ipmaddr`	`ip maddr`
Statistiques	`netstat`	`ss`

1. `net-tools` n'est plus installé par défaut avec la version Desktop d'Ubuntu 18.10, mais l'est toujours dans la version Server

Lister les interfaces

- jusqu'à la couche 2 / liaison (adresses MAC) : `ip l[ink] [show]`)
- jusqu'à la couche 3 / réseau (adresses IP) : `ip a[ddr] [show]`)

```
$ ip -4 a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
UNKNOWN group default qlen 1000
    inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
fq_codel state UP group default qlen 1000
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
      valid_lft 79785sec preferred_lft 79785sec
```

1. Si on veut seulement le *nom* des interfaces, on peut utiliser `networkctl`
2. Pour avoir des infos constructeur : `lshw -class network` (lshw = *list hardware*)

Lister les interfaces

```
$ ip -4 a
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
fq_codel state UP group default qlen 1000
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 79785sec preferred_lft 79785sec
```

- `<BROADCAST,MULTICAST,UP,LOWER_UP>` : la carte peut faire du broadcast (et donc du multicast), est active (`UP`) et le câble est branché (`LOWER_UP`)
- `mtu 1500` : taille maximale d'une unité de transfert
- `qdisc fq_codel` : algorithme d'ordonnancement de l'envoi des paquets
- `state UP` : réseau opérationnel, il y a une connexion
- `qlen 1000` : taille maximale de la file de transmission

1. Documentation : `http://linux-ip.net/gl/ip-cref/ip-cref-node17.html` et `https://www.howtogeek.com/657911/how-to-use-the-ip-command-on-linux/amp/`

Configurer une interface réseau

Activer une interface : `ip link set enp0s3 up`

Désactiver une interface : `ip link set enp0s3 down`

Attribuer d'une adresse IP automatique par DHCP : `dhclient enp0s3`

Attribuer une adresse IP : `ip addr add 192.168.1.100/24 dev enp0s3`

Supprimer une adresse IP : `ip addr del 192.168.1.100/24 dev enp0s3`

Supprimer toute la configuration d'une interface : `ip addr flush enp0s3`

La configuration avec ces outils est `temporaire`!

Netplan

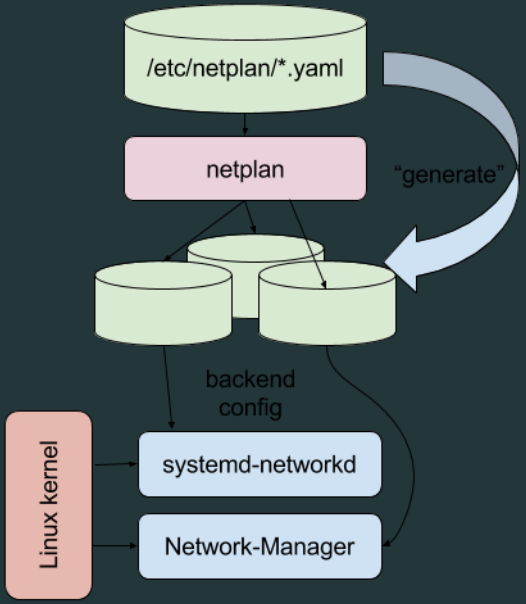
Avant Ubuntu 17.10, on configurait un réseau avec le paquet `ifupdown` et le fichier `/etc/network/interfaces`.

Depuis Ubuntu 17.10 : Netplan (fichiers de configuration au format YAML stockés dans `/etc/netplan`[1]).

Version	Renderer	Fichier
Desktop	NetworkManager	`01-network-manager-all.yaml`
Server	networkd	`01-netcfg.yaml`
Cloud	networkd	`50-cloud-init.yaml`

1. Par ordre d'importance, on peut trouver ces fichiers dans `/lib/netplan`, `/etc/netplan` et `/run/netplan`. Les fichiers sont traités dans l'ordre numérique, et un fichier remplace la configuration d'un fichier précédent pour une même interface

Netplan



`netplan generate` génère les fichiers de configuration dans `/run/systemd/network`

Netplan

2 types de renderer :

- `networkd` (surtout version Server / valeur par défaut)
- `NetworkManager` (surtout version Desktop)

Quelques commandes à connaître :

- `netplan try` : tester une configuration
- `netplan [--debug] apply` : appliquer une configuration
- `systemctl restart systemd-networkd` : redémarrer le service réseau

```
network :
  version : 2
  renderer : networkd
  ethernets :
    enp0s3 :
      dhcp4 : true
```

⚠ Les indentations se font avec des espaces (aucune tabulation)

Netplan / Exemple : attribuer une adresse IP statique

```
network :
  version : 2
  renderer : networkd
  ethernets :
    enp0s3 :
      addresses :
        - 192.168.100.1/24
```

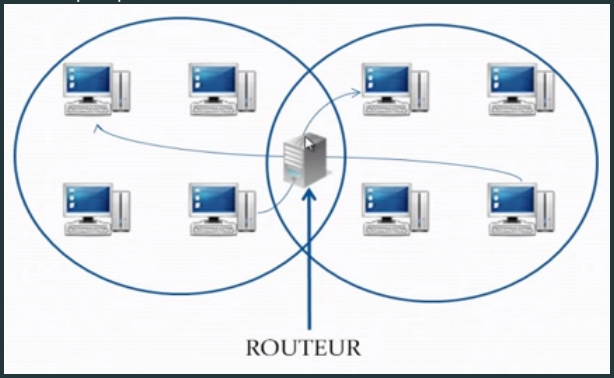
```
network:
  version: 2
  renderer: networkd
  wifis:
    wlp2s0b1:
      dhcp4: yes
      access-points:
        "SSID_du_WiFi":
          password: "**********"
```

```
network :
  version : 2
  renderer : networkd
  ethernets :
    enp0s25 :
      addresses :
        - 192.168.0.100/24
      gateway4 : 192.168.0.1
      nameservers :
          search : [exemple.org, cpe.fr]
          addresses : [1.1.1.1, 8.8.8.8, 4.4.4.4]
```

Le champ `search` permet de spécifier des noms de domaines utilisés pour la résolution de nom : si je tape `machine` dans un navigateur, il essaiera de contacter successivement `machine.exemple.org.` et `machine.cpe.fr.`

Routage

Rappel : routeur = machine servant d'interface entre des réseaux (*passerelle*) et assurant le transit des paquets



Un routeur a autant d'interfaces réseau que de réseaux auxquels il est connecté. Quand un routeur est connecté à plus de deux réseaux, on utilise une table de routage pour savoir où envoyer un paquet.

Routage

Visualiser la table de routage : `ip r[oute] [list]`

```
$ ip r
default via 10.0.2.2 dev enp0s3 proto dhcp src 10.0.2.15 metric
100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15
10.0.2.2 dev enp0s3 proto dhcp scope link src 10.0.2.15 metric 100
```

- `default via 10.0.2.2` : adresse du routeur par défaut
- `dev enp0s3` : carte réseau utilisée
- `proto dhcp / kernel` : protocole de routage (dynamique / auto-configuré)
- `metric 100` : niveau de préférence (plus une métrique est faible, plus la route est préférée); utile par exemple pour donner une préférence sur une connexion filaire plutôt que Wi-Fi.

Ajouter une passerelle :

```
ip route add default via 192.168.1.1
```

Outils réseau

ping

La commande `ping <IP> | <nom d'hôte>` permet de tester l'accessibilité d'une machine :

```
$ ping fr.wikipedia.org
PING r, r.knams.wikimedia.org (145.97.39.137): 56 data bytes
64 bytes from 145.97.39.137: icmp_seq=0 ttl=53 time=51.9 ms
64 bytes from 145.97.39.137: icmp_seq=1 ttl=53 time=16.9 ms
64 bytes from 145.97.39.137: icmp_seq=2 ttl=53 time=16.7 ms

--- rr.knams.wikimedia.org ping statistics ---
3 packets transmitted, 3 packets received, 0 % packet loss
round-trip min/avg/max = 16.7/28.5/51.9 ms
```

host

La commande `host` permet d'effectuer des requêtes DNS, notamment pour convertir des noms d'hôte en adresse IP et réciproquement :

```
$ host www.wikipedia.fr
www.wikipedia.fr has address 78.109.84.114

$ host 78.109.84.114
114.84.109.78.in-addr.arpa domain name pointer
wikimedia2.typhon.net.
```

host

On peut aussi obtenir les serveurs DNS qui gèrent un domaine :

```
$ host -t NS wikipedia.fr
wikipedia.fr name server b.dns.gandi.net.
wikipedia.fr name server c.dns.gandi.net.
wikipedia.fr name server a.dns.gandi.net.
```

Ou les serveurs de messagerie [1] pour ce domaine :

```
$ host -t MX wikipedia.fr
wikipedia.fr mail is handled by 50 fb.mail.gandi.net.
wikipedia.fr mail is handled by 10 spool.mail.gandi.net.
```

`host` affiche une réponse par ligne et est donc bien adapté aux scripts ; il est approprié pour des réponses simples et rapides

1. Les nombres correspondent aux priorités ; le plus petit nombre a la plus grande priorité

dig

La commande dig permet de réaliser des tâches similaires[1] :

```
$ dig wikipedia.fr + short
78.109.84.114
$ dig NS wikipedia.fr + short
a.dns.gandi.net.
c.dns.gandi.net.
b.dns.gandi.net.
$ dig MX wikipedia.fr + short
50 fb.mail.gandi.net.
10 spool.mail.gandi.net.
$ dig -x 8.8.8.8 +short
google-public-dns-a.google.com.
```

1. Sans l'option + short, dig fournit des informations souvent plus complète que host

Sécurité du réseau : Netfilter / Iptables

Netfilter : présentation

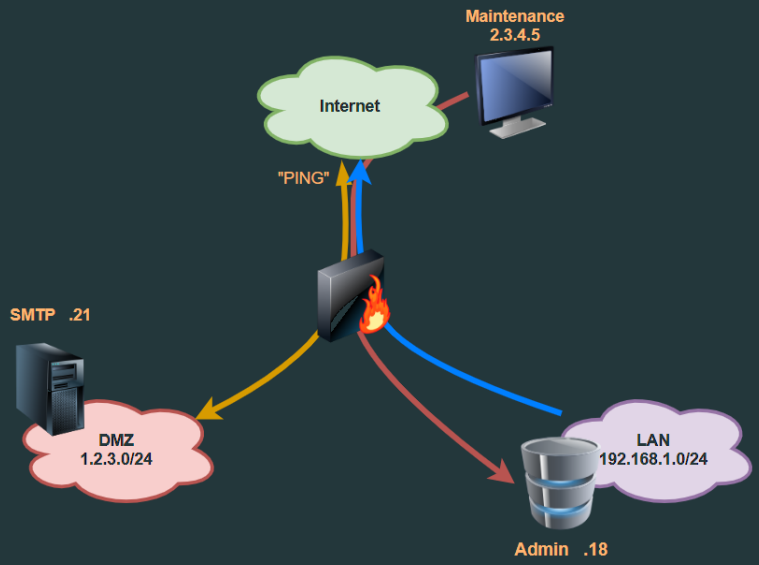
Module du noyau Linux permettant de filtrer et manipuler les paquets réseau qui passent dans le système. Il fournit :

- un pare-feu (contrôle des connexions des machines, sur quels ports…)
- de la traduction d'adresse (NAT) pour partager une connexion (*masquerading*), masquer des machines du réseau local, rediriger des connexions…
- une journalisation du trafic réseau

Les paquets sont interceptés (à la réception, avant de les transmettre aux processus, avant de les envoyer à la carte réseau, etc.), et passent à travers des chaînes de règles qui déterminent ce que le système doit en faire (bloquer, laisser passer…).

Netfilter : exemple

Considérons la situation suivante :



Netfilter : exemple

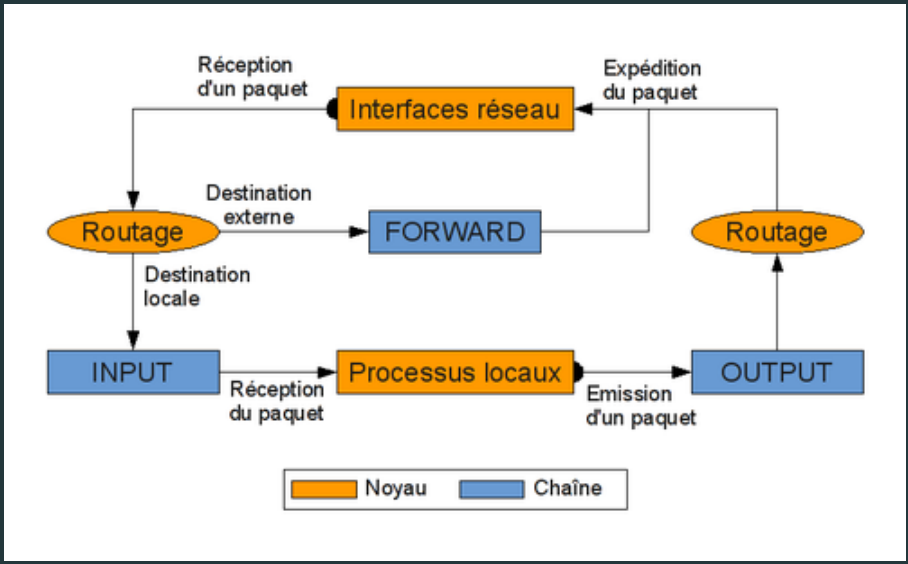
4 règles à configurer pour ce pare-feu :

1. Autoriser les utilisateurs du LAN à accéder à Internet
2. Autoriser tout le monde à accéder au serveur mail
3. Autoriser le pare-feu à pinguer sur Internet
4. Autoriser une exception pour accéder au serveur sur le LAN

3 catégories de règles de filtrage :

- filtrage des paquets à transmettre (Règles 1 et 2) => chaîne `FORWARD`
- filtrage des paquets émis par le système (Règle 3) => chaîne `OUTPUT`
- filtrage des paquets arrivant dans le système (Règle 4) => chaîne `INPUT`

Cheminement des paquets à travers Netfilter :



Netfilter : chaînes de règles

Une *chaîne* est un ensemble de règles qui indiquent ce qu'il faut faire des paquets qui la traversent.

Une *règle* est une combinaison de *critères* et d'une *cible*.

Principe des chaînes de règles

- Les règles sont lues dans l'ordre
- Une règle est remplie si tous ses critères sont satisfaits
- Dès qu'une règle est remplie, la cible [1] est exécutée, et les règles suivantes sont ignorées
- Si aucune règle n'a interrompu le parcours de la chaîne, une politique par défaut est appliquée

1. `ACCEPT`, `REJECT`, `DROP` (rejet sans notification à l'émetteur), `LOG`...

Netfilter

Netfilter est un *framework*; il s'utilise *via* des utilitaires :

- iptables : l'utilitaire de longue date pour Netfilter
- nftables : le remplaçant annoncé d'iptables
- ufw (uncomplicated firewall) : alternative simplifiée à iptables

Iptables

`iptables -L` : liste les chaînes de règles courantes
`iptables -A <chaîne>` *règle* : ajoute une règle à la fin de la chaîne
`iptables -I <chaîne>` *règle* `<i>` : ajoute une règle en i-ème position
`iptables -D` : supprime la règle spécifiée
`iptables -F [chaîne]` : supprime toutes les règles [de la chaîne] ⚠
`iptables -P` : modifie la politique d'une chaîne (règle par défaut)

Exemple : pare-feu non configuré, tout le trafic est autorisé (`policy ACCEPT`) :

```
$ sudo 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
```

Exemple

Pour rejeter tous les paquets en provenance de `192.168.1.11` :

```
iptables -A INPUT -s 192.168.1.11 -j DROP
```

Ici, le *critère* est une adresse source ; d'autres critères possibles :

- `-d` : un adresse de destination
- `-p` : un protocole (`tcp, udp, icmp...`)[1]
- `-sport` : un port source
- `-dport` : un port destination
- `-i` : l'interface réseau d'où provient le paquet
- `-o` : l'interface réseau de destination du paquet

1. Voir `/etc/protocols` pour les autres protocoles

Autres exemples

Pour interdire les entrées par `enp0s3` :

```
iptables -A INPUT -i enp0s3 -j DROP
```

Pour interdire le protocole ICMP (*ping*) en entrée :

```
iptables -A INPUT -p icmp -j DROP
```

Pour interdire les connexions entrantes à destination du port 80 :

```
iptables -A INPUT -p tcp --dport 80 -j DROP
```

Pour interdire toutes les connexions sauf celle de 10.0.0.1 :

```
iptables -A INPUT -s ! 10.0.0.1 -j DROP
```

Pour loguer les événements :

```
iptables -A INPUT -m limit --limit 5/min -j LOG
--log-prefix "iptables denied: " --log-level 7
```

Iptables : tables

Les chaînes sont regroupées en tableaux (ou *tables*). Les 3 principales :

- `filter` (tableau par défaut) : chaînes de filtrage pour accepter, refuser, ignorer un paquet (chaînes INPUT, FORWARD, OUTPUT)
- `nat` : chaînes de modification des adresses IP ou des ports sources ou destinataires (chaînes PREROUTING, OUTPUT, POSTROUTING)
- `mangle` : chaînes permettant de modifier certains paramètres à l'intérieur des paquets IP (chaînes PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING)

Ex. : *MASQUERADING* (= NAT source) : autoriser les machines avec une IP privée à accéder à Internet

```
$ iptables -t nat -a POSTROUTING -s 192.168.1.0/24 -o enp0s3 -j MASQUERADE
```

Important

Les règles sont transmises dynamiquement au noyau, et sont perdues au redémarrage de la machine !

Plusieurs solutions :

- renseigner les règles dans un script exécuté au démarrage
- utiliser le paquet `iptables-persistent` (crée les fichiers `/etc/iptables/rules.v[4|6]`, lus au démarrage)

UFW : Uncomplicated Firewall

Front-end pour NetFilter

`ufw enable / disable`	Active / Désactive le pare-feu
`ufw status [verbose]`	Affiche le statut du pare-feu
`ufw allow / deny [règle]`	Autorise / Refuse une connexion
`ufw logging on / off`	Active / Désactive la journalisation
`ufw app list`	Liste les services qui ont des règles `ufw`
`ufw app info APP`	Affiche les règles de *APP*
`ufw [--dry-run] règle`	Affiche les changements impliqués par *règle* sans les appliquer

UFW : Uncomplicated Firewall

Exemples :

- `ufw default allow outgoing` : autorise toutes les connexions sortantes
- `ufw default deny ingoing` : bloque toutes les connexions entrantes
- `ufw deny 80` : bloque le port 80
- `ufw deny http` : bloque le service HTTP
- `ufw allow proto tcp from 1.2.3.4 to 5.6.7.8 port 22` : autorise la seule IP `1.2.3.4` à se connecter au serveur `5.6.7.8` par ssh

Sécurité des services

Notion de socket

Rappel

Une machine dispose d'une adresse IP et de ports de connexion (virtuels), numérotés, auxquels sont rattachés des services (cf. `/etc/services`).

Un client établit une connexion depuis un port de sa machine vers le port d'une autre machine (typiquement, le port 80 d'un serveur web). Le canal de communication établi entre ces deux machines est ce qu'on appelle un[1] socket.

Important

Sans pare-feu, les ports ne sont pas filtrés et n'importe qui peut essayer d'ouvrir un socket.

⇒ sans danger sur un port fermé

⇒ mais risque d'intrusion sur un port ouvert !

1. On trouve parfois aussi *une* socket

Identifier les ports ouverts

Il existe de nombreux outils pour surveiller les ports :

- `netstat`, l'un des plus connus (mais n'est plus installé par défaut)
- `ss` (*socket statistics*), plus récent (package `iproute2`)
- `netcat` / `nc`, qui permet de faire davantage que de la surveillance de ports
- `lsof -i` *(list open files)* sous Linux, les sockets sont des fichiers !

L'outil `netstat` permet de vérifier quels sont les ports à l'écoute sur la machine, qui a établi une connexion et quels sont les services locaux à l'écoute

Balayer les ports avec nmap

Célèbre[1] scanneur de ports, très complet

Il est notamment capable de déterminer, sur la machine cible les services actifs, leur version, ainsi que celle du système d'exploitation :

Attention !

Le balayage de ports peut être considéré comme illégal. Il ne doit donc être utilisé que par un administrateur sur son propre réseau pour vérifier sa sécurité.

1. Il a notamment joué dans *Matrix*, *Die Hard 4*, *G.I. Joe*, *La vengeance dans la peau*, ...

Balayer les ports avec nmap

Exemple :

```
$ sudo nmap -O localhost
Starting Nmap 7.60 ( https://nmap.org )
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000015s latency).
Not shown: 998 closed ports
PORT STATE SERVICE
80/tcp open http
631/tcp open ipp
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3
OS details: Linux 3.7 - 3.10
Network Distance: 0 hops
```

Quand utiliser `netstat` ou `nmap` ?

`netstat` :

- renseigne sur les ports à l'écoute sur la machine *locale*
- donne des informations pour chaque port (service à l'écoute)
- tire ses informations de l'OS

`nmap` :

- peut renseigner sur les ports à l'écoute sur une machine *distance* (attention aux implications légales)
- peut donner des informations plus poussées sur les services ou l'OS (comme les numéros de version)
- tire ses informations des paquets reçus ; peut être utile si on soupçonne la présence d'un rootkit qui corrompt les résultats de *netstat*