

La sécurité réseau

Objectifs de cette section

Apprendre des **techniques** et **stratégies** pour sécuriser les services réseaux.

Comprendre comment le **pare-feu** local fonctionne sur Linux et comment le configurer.

Prévention de la **fuite d'informations**.

Apprendre comment tester les **ports ouverts** sur la machine et effectuer des scans de ports.

A quoi sert **xinetd**, ce qu'il fait et comment le sécuriser.

Comment sécuriser le protocole **SSH**

Sécurité des services réseaux

Les services réseaux

Ils sont appelés services, daemons, ou serveurs.

Ces services écoutent sur des ports réseaux pour des connexions entrantes (par exemple le port 80 pour le service web).

Imaginez que votre système est comme une maison, et que chaque fenêtre ouverte correspond à un port sur lequel un service écoute (c'est donc une entrée possible pour les attaquants).

Ils fonctionnent en arrière-plan de manière constante.

Les services réseaux (2)

Les sorties standards de ces services sont généralement enregistrés sous la forme de log (par exemple pour chaque connexion via SSH qui écoute sur le port 22) stockés sur votre système dans le dossier `/var/log` la plupart du temps.

Les services sont conçus initialement pour n'effectuer qu'une seule tâche.

- Le daemon SSHd écoute sur le port 22 pour l'établissement d'une connexion SSH
- Le daemon httpd écoute sur le port 80 pour répondre aux requêtes web

Sécurisation des services réseaux

Il faut toujours utiliser un compte dédié pour chaque service.

- Ainsi, si ce service est compromis par une attaque, l'attaquant aura plus de difficultés à obtenir des privilèges pour l'exécution de tâches communes.
- Tire partie de la séparation des privilèges.

Les ports inférieurs à 1024 sont dits « réservés » (Well Known Ports)

- Nécessitent l'utilisation du compte root pour être ouverts.
- Sont ensuite utilisés par les utilisateurs propres à chaque service.

Il faut toujours arrêter et désinstaller les services qui ne sont plus utilisés (comme lorsqu'on ferme les fenêtres de chez soi si nous ne sommes plus là).

Sécurisation des services réseaux (2)

Eviter l'utilisation de services qui transmettent et reçoivent des données de manière non chiffrée.

- Préférer le SSH au Telnet
- Préférer l'HTTPS à l'HTTP

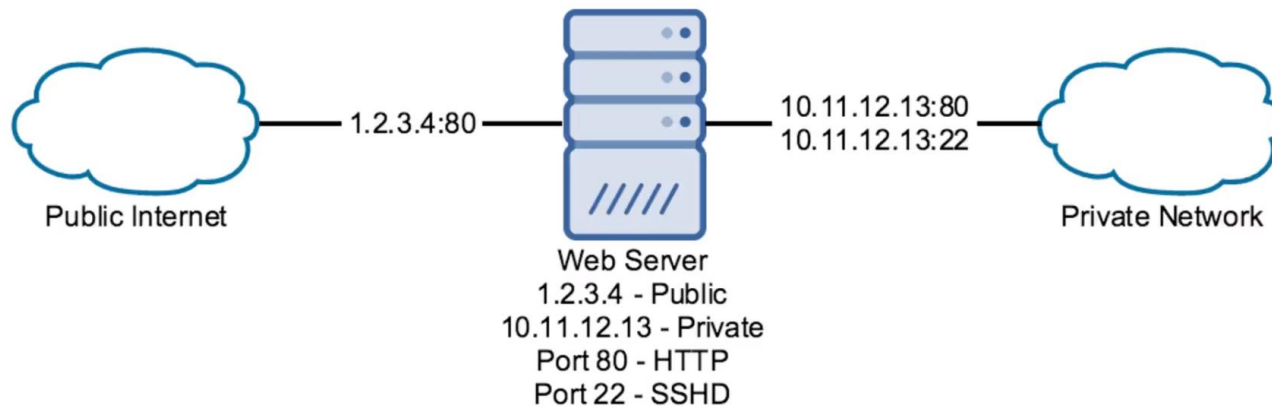
Garder vos services à jour en installant les patches de sécurité.

- Préférer les services fournis par votre distribution (via aptitude) plutôt que ceux trouvés sur Internet pour faciliter l'installation des mises à jour.

Demander à vos services de n'écouter que sur les interfaces et les adresses IP nécessaires.

Sécurisation des services réseaux (3)

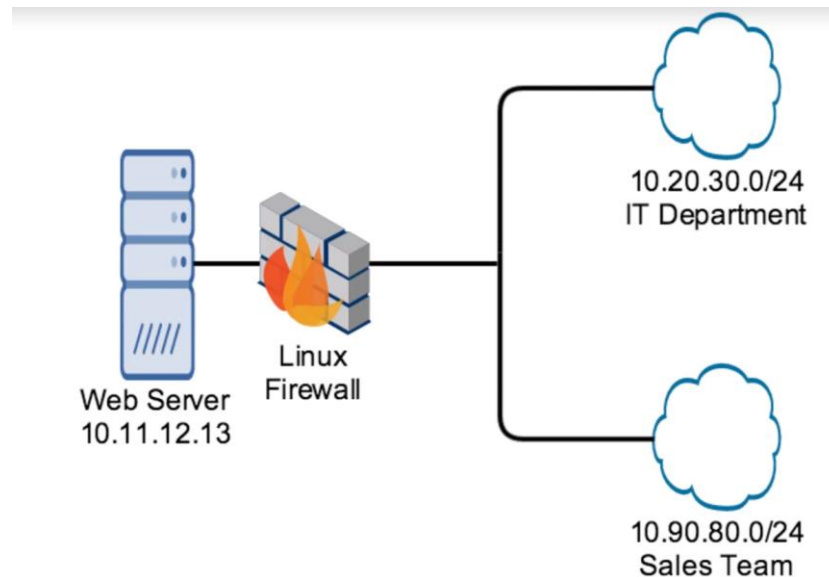
Demander à vos services de n'écouter que sur les interfaces et les adresses IP nécessaires.



Ici rien ne sert de laisser les connexions SSH depuis Internet, car elles ne seront émises que depuis votre réseau privé.

Sécurisation des services réseaux (3)

Vous pouvez également utiliser le pare-feu personnel de Linux pour effectuer un filtrage.



Ici on peut demander au pare-feu de ne laisser les connexions SSH passer que si elles sont émises depuis les adresses IP correspondantes aux machines du personnel informatique de l'entreprise.

Les fuites d'informations

Eviter de divulguer des informations sur vos services lorsque c'est possible.

- Par exemple la version d'apache que votre site web utilise.
- Visible grâce à la commande : `curl -I http://IP DU SERVEUR`

```
root@debian:/home/jordan# curl -I http://127.0.0.1
HTTP/1.1 200 OK
Date: Wed, 20 Jun 2018 15:10:15 GMT
Server: Apache/2.4.25 (Debian)
Last-Modified: Wed, 20 Jun 2018 15:10:01 GMT
ETag: "29cd-56f143152b262"
Accept-Ranges: bytes
Content-Length: 10701
Vary: Accept-Encoding
Content-Type: text/html
```

Les fuites d'informations (2)

Attention, certaines informations sur votre système sont disponibles pour tous les utilisateurs :

```
jordan@debian:~$ cat /etc/issue  
Debian GNU/Linux 9 \n \l
```

```
jordan@debian:~$ cat /etc/issue.net  
Debian GNU/Linux 9
```

Afficher les services utilisés par votre système avec **systemctl**

La commande **systemctl** permet d'afficher les services utilisés par votre système :

<code>apache2.service</code>	<code>loaded active running</code>	<code>The Apache HTTP Server</code>
<code>ssh.service</code>	<code>loaded active running</code>	<code>OpenBSD Secure Shell [...]</code>
<code>apt-daily-upgrade.timer</code>	<code>loaded active waiting</code>	<code>Daily apt upgrade [...]</code>

Stopper et désactiver des services

Si votre système utilise le `systemctl`, alors vous pouvez utiliser les commandes :

- `systemctl stop NOM_DU_SERVICE`
- `systemctl disable NOM_DU_SERVICE`

Sur Debian, vous pouvez également utiliser, au choix :

- `service NOM_DU_SERVICE stop`
- `/etc/init.d/NOM_DU_SERVICE stop`

Utiliser la commande netstat

Netstat est une commande vous permettant de savoir quels sont les services qui écoutent spécifiquement sur des ports :

```
root@debian:/home/jordan# netstat -nulp
```

```
Active Internet connections (only servers)
```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN	410/sshd
tcp	0	0	127.0.0.1:631	0.0.0.0:*	LISTEN	1043/cupsd
udp	0	0	0.0.0.0:631	0.0.0.0:*		1045/cups-browsed
udp	0	0	0.0.0.0:1900	0.0.0.0:*		713/minissdpd
udp	0	0	0.0.0.0:68	0.0.0.0:*		403/dhclient

Le scan de ports

Il est également possible d'effectuer un scan des ports ouverts grâce à la commande **nmap** :

```
root@debian:/home/jordan# nmap 127.0.0.1

Starting Nmap 7.40 ( https://nmap.org ) at 2018-06-20 16:28 BST

Nmap scan report for localhost (127.0.0.1)

Host is up (0.0000060s latency).

Not shown: 997 closed ports

PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
631/tcp   open  ipp

Nmap done: 1 IP address (1 host up) scanned in 1.63 seconds
```

Tester un port spécifique

Il est possible de tester spécifiquement un port grâce à la commande **telnet** :

```
root@debian:/home/jordan# telnet 127.0.0.1 22
```

```
Trying 127.0.0.1...
```

```
Connected to 127.0.0.1.
```

```
Escape character is '^]'.
```

```
SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u3
```

Tester un port spécifique (2)

Il est possible de tester spécifiquement un port grâce à la commande **nc -v** :

```
root@debian:/home/jordan# nc -v 127.0.0.1 22
localhost [127.0.0.1] 22 (ssh) open
SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u3
```

Chiffrement asymétrique des données

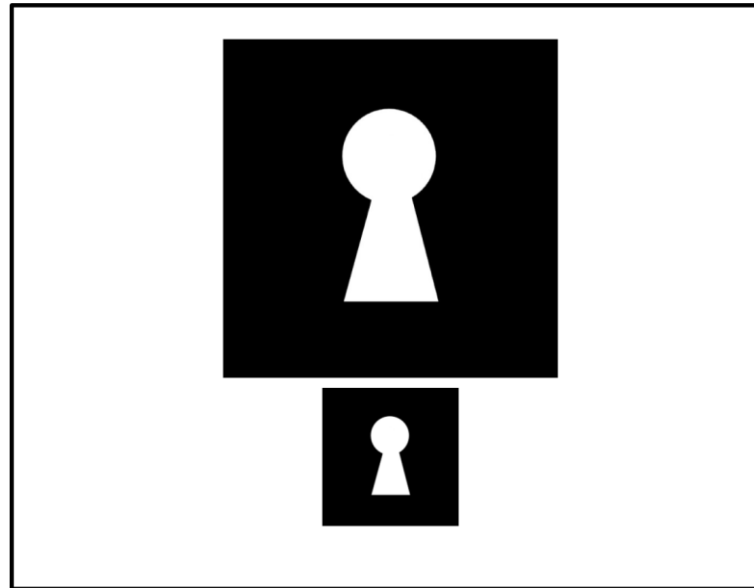
Qu'est-ce que c'est ?

Un bon exemple du chiffrement asymétrique correspond au système **de clé publique et clé privée**.

En effet, plutôt que d'utiliser la même clé pour le chiffrement et le déchiffrement, on utilise deux clés différentes qui fonctionnent mathématiquement sous forme de paire.

Clé publique et clé privée

Pour bien comprendre l'interaction entre clé privée et clé publique, imaginez un gros coffre comportant deux serrures, une grande et une plus petite.



Clé publique et clé privée

Le coffre contient les données que l'on souhaite transmettre de manière chiffrée à notre destinataire.

Lorsque les données sont mises dans le coffre, alors on le **verrouille à l'aide de la clé publique** (grosse serrure).

Cependant, il ne sera possible de rouvrir le coffre que si on le **déverrouille grâce à la clé privée** rentrant dans la petite serrure.

De la même manière, **si le coffre est verrouillé avec la clé privée, alors il ne pourra être déverrouillé que grâce à la clé publique** qui lui est associée.

On comprend donc que ces deux clés fonctionnent toujours par paire mathématiquement liée.

Limitations

Attention car il y a une grosse consommation de **CPU** lors de l'utilisation de la paire de clés pour chiffrer ou déchiffrer les données.

Pour cette raison, les algorithmes de chiffrement asymétrique sont utilisées de façon parcimonieuse. En effet, plutôt que d'utiliser ces algorithmes pour chiffrer la totalité de nos données, on les **utilise plutôt pour effectuer de l'authentification** d'un peer VPN ou la génération de matériel de chiffrement qui pourra être utilisé par les algorithmes symétriques.

Pourquoi les noms privées et publics ?

L'utilisation des mots publique et privée pour faire référence aux clés ne sont pas anodins.

En effet, la clé publique est justement rendue publique et disponible pour toutes les personnes qui souhaite l'utiliser.

Par contre, la clé privée, comme son nom l'indique, n'est connue que de l'équipement propriétaire de la clé publique associée.

Quelques algorithmes de chiffrement asymétrique

RSA : appelé ainsi parce que développé par Rivest, Shamir et Adleman. Le RSA est surtout utilisé aujourd'hui pour l'authentification. La longueur de la clé peut aller de 512 à 2048 bits, même si on conseille de prendre au minimum une clé de longueur 1024 bits.

ElGamal : Système de chiffrement asymétrique basé sur un échange de type DH.

DSA (Digital Signature Algorithm) : développé par la NSA américaine.

ECC (Elliptic Curve Cryptography)

Sécuriser le SSH

Sécurisation du SSH

SSH est un protocole signifiant **Secure Shell**.

- Permet d'établir une connexion sécurisée à travers un réseau.

SSH permet d'établir une connexion à l'aide d'un nom d'utilisateur et d'un mot de passe, mais également en utilisant des clés d'authentification.

- Vérifier dans le fichier **/etc/ssh/sshd_config** que la ligne suivante est présente :
- `PubKeyAuthentication yes`

Créer des clés SSH

Sur Linux on utilise la commande **ssh-keygen** pour créer ses clés.

Avec cette commande, le système va vous demander si vous voulez protéger le contenu de la clé à l'aide d'un mot de passe (passphrase).

Cette commande crée deux fichiers :

- `~/.ssh/id_rsa` : clé privée
- `~/.ssh/id_rsa.pub` : clé publique

Ajouter la clé publique à un hôte distant

Pour envoyer la clé publique sur un hôte distant (qui va chiffrer les données vous étant destinées avec) on utilise la commande :

- `ssh-copy-id nom_utilisateur@adresse_ip_distante`

Cette clé va être stockée dans le fichier `~/.ssh/authorized_keys` de l'hôte distant.

Ainsi, vous serez le seul à être capable de déchiffrer les données, puisque vous seul possédez la clé privée associée à cette clé publique.

Forcer l'utilisation des clés pour l'authentification

Pour interdire l'authentification avec utilisateur et mot de passe, et n'autoriser que les authentifications avec les clés, il suffit de mettre la ligne suivante dans le fichier **/etc/ssh/sshd_config** :

- `PasswordAuthentication no`

Renforcer la connexion en root via SSH

Pour désactiver la possibilité de se connecter à la machine directement en Root, toujours dans le fichier `/etc/ssh/sshd_config`, mettre la ligne suivante :

- `PermitRootLogin no`

Pour n'autoriser la possibilité de se connecter en Root qu'avec une clé, mettre la ligne suivante :

- `PermitRootLogin without-password`

Renforcer la connexion des autres utilisateurs

Il est également possible de ne restreindre l'accès à la machine en SSH que pour certains comptes utilisateurs :

- `AllowUsers utilisateur1 utilisateur2 utilisateur3 ...`

Il est également possible de ne restreindre l'accès à la machine en SSH que pour certains groupes d'utilisateurs :

- `AllowGroups groupe1 groupe2 groupe3 ...`

De la même manière, vous pouvez interdire l'accès SSH à certains utilisateurs ou groupes :

- `DenyUsers utilisateur1 utilisateur2 utilisateur3 ...`
- `DenyGroups groupe1 groupe2 groupe3 ...`

Pare-Feu Linux

NETFILTER ET IPTABLES

Le pare-feu Linux

Un pare-feu est conçu pour bloquer les accès réseaux non autorisés, et autoriser ceux qui le sont.

Le pare-feu Linux est composé de deux éléments :

- Netfilter : framework à l'intérieur du noyau Linux
- IPTables : système de sélection des paquets réseaux
 - Utilisation de la commande `iptables`

TABLE 1

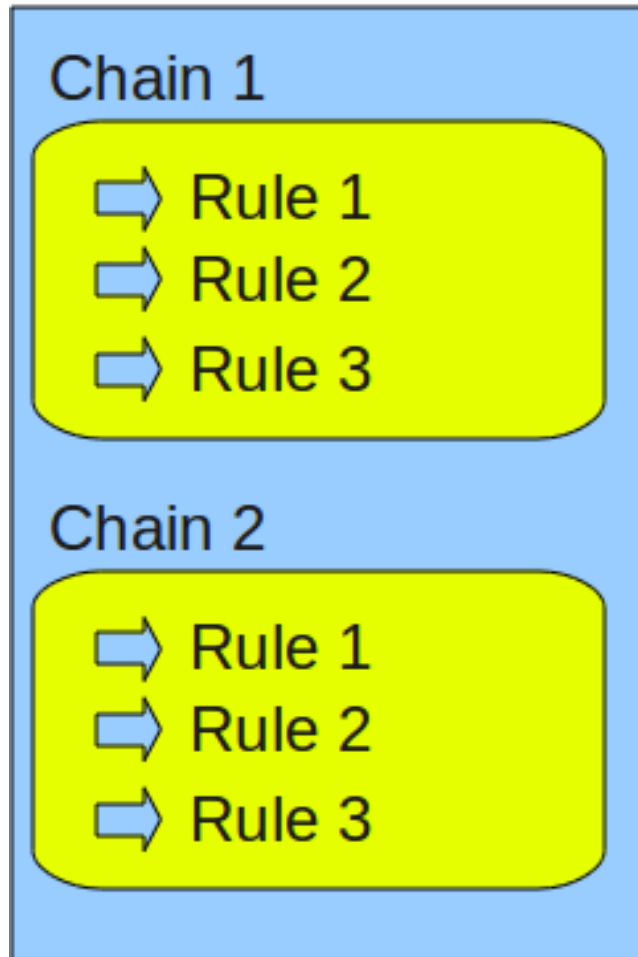
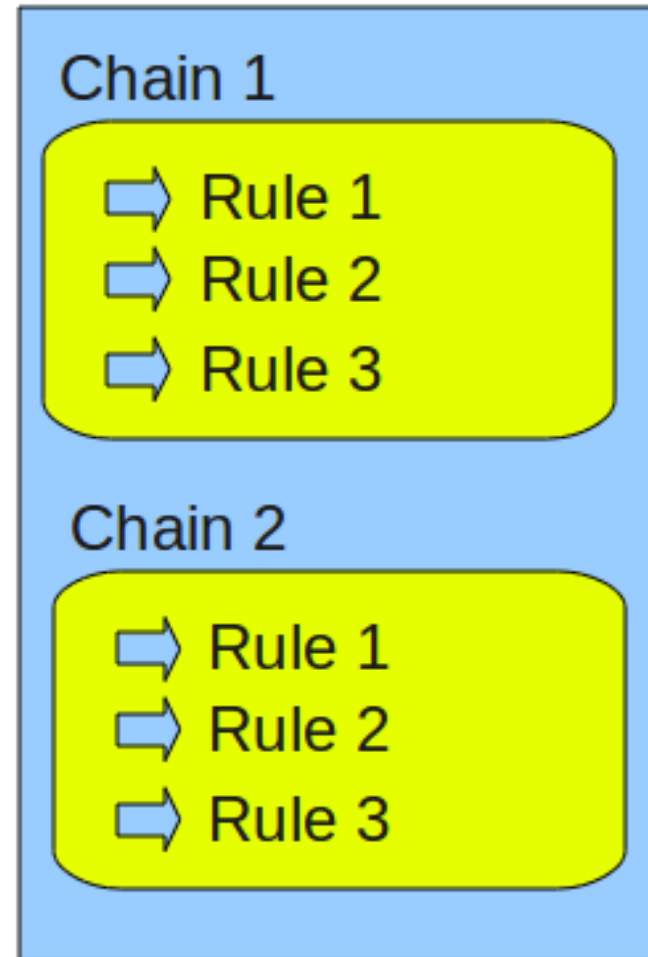


TABLE 2



Les tables par défaut

Filter : Table par défaut, la plus utilisée.

- Permet de bloquer ou autoriser des demandes de connexion entrantes ou sortantes.

NAT : Utilisée pour la translation d'adresse (Network Address Translation)

MANGLE : Utilisée pour altérer les paquets (modification du TTL, ou des autres champs existants)

RAW : Peu utilisée, mais permet de désactiver le suivi des connexions

SECURITY : Utilisée par le SELinux qui est système de contrôlé d'accès.

- En pratique le noyau Linux interroge SELinux avant chaque appel système pour savoir si le processus est autorisé à effectuer l'opération donnée.

Les chaines par défaut

INPUT

OUTPUT

FORWARD

PREROUTING

POSTROUTING

Ps : Ces chaînes ne contiennent aucune règle par défaut.

	INPUT	OUTPUT	FORWARD	PREROUTING	POSTROUTING
Filter	X	X	X		
Nat	X	X		X	X
Mangle	X	X	X	X	X
Raw		X		X	
Security	X	X	X		

Table: filter

Chain: INPUT

Rules

Chain: FORWARD

Rules

Chain: OUTPUT

Rules

Table: nat

Chain: PREROUTING

Rules

Chain: INPUT

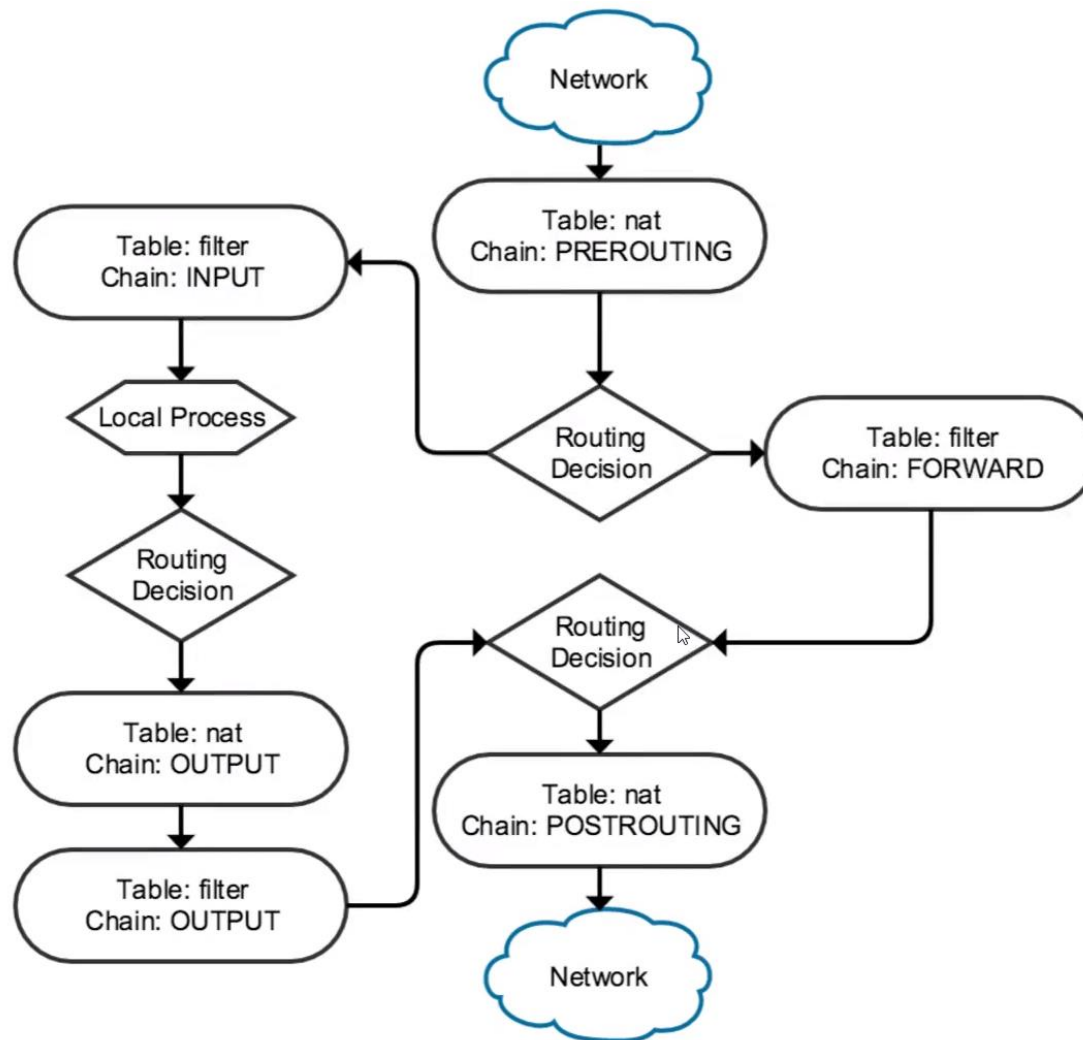
Rules

Chain: OUTPUT

Rules

Chain: POSTROUTING

Rules



Règles

Les règles sont composées principalement de deux éléments :

- Match
- Target

Le matching peut se faire sur :

- Protocole
- IP ou Réseau source et destination
- Port source et destination
- Interface Réseau
- Exemple : protocol: TCP, IP source: 192.168.1.2,dest port: 80

Le target

Le target permet de préciser au système ce qu'il doit faire avec les paquets qui ont matché !

On peut pour cela utiliser une chaîne, ou directement les possibilités suivantes :

- ACCEPT : accepte le paquet et n'examine pas la règle suivante.
- DROP : ignore le paquet de manière silencieuse et n'examine pas la règle suivante.
- REJECT : ignore le paquet en envoyant une notification de rejet et n'examine pas la règle suivante.
- LOG : permet d'enregistrer un log correspondant au paquet qui a matché et passe à la règle suivante.
- RETURN

Implémentation du firewall Linux

Les commandes disponibles

`iptables -L` : Affiche la table filter.

`iptables -t nat -L` : Affiche la table NAT.

`iptables -nL` : Affiche la table filter en utilisant une sortie « numérique ».

`iptables -vL` : Affiche la table filter en affichant plus de détails.

`iptables --line-numbers -L` : Affiche la table filter en utilisant des numéros de lignes.

jordan@debian: ~/Desktop

- □ ×

File Edit View Search Terminal Help

```
root@debian:/home/jordan/Desktop# 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
```

```
root@debian:/home/jordan/Desktop#
```

```
# iptables -nL
```

```
Chain INPUT (policy DROP)
```

target	prot	opt	source	destination	
DROP	all	--	216.58.219.174	0.0.0.0/0	
ACCEPT	tcp	--	0.0.0.0/0	0.0.0.0/0	tcp dpt:80
ACCEPT	tcp	--	0.0.0.0/0	0.0.0.0/0	tcp dpt:433
ACCEPT	tcp	--	10.11.12.0/24	0.0.0.0/0	tcp dpt:22
ACCEPT	icmp	--	10.11.12.0/24	0.0.0.0/0	icmptype 8

```
Chain FORWARD (policy ACCEPT)
```

target	prot	opt	source	destination
--------	------	-----	--------	-------------

```
Chain OUTPUT (policy ACCEPT)
```

target	prot	opt	source	destination
--------	------	-----	--------	-------------

La police de chaîne / Comportement par défaut

Il est possible de donner à la chaîne l'action à effectuer si le paquet ne matche avec aucunes des règles spécifiées à l'intérieur de cette chaîne.

On utilise alors la commande :

- `iptables -P CHAIN TARGET`

Par exemple, pour dire que sur la chaîne INPUT, si le paquet ne matche avec aucune règle, on souhaite le rejeter :

- `iptables -P INPUT DROP`

```
# iptables -nL
```

```
Chain INPUT (policy DROP)
```

target	prot	opt	source	destination	
DROP	all	--	216.58.219.174	0.0.0.0/0	
ACCEPT	tcp	--	0.0.0.0/0	0.0.0.0/0	tcp dpt:80
ACCEPT	tcp	--	0.0.0.0/0	0.0.0.0/0	tcp dpt:433
ACCEPT	tcp	--	10.11.12.0/24	0.0.0.0/0	tcp dpt:22
ACCEPT	icmp	--	10.11.12.0/24	0.0.0.0/0	icmptype 8

```
Chain FORWARD (policy ACCEPT)
```

target	prot	opt	source	destination
--------	------	-----	--------	-------------

```
Chain OUTPUT (policy ACCEPT)
```

target	prot	opt	source	destination
--------	------	-----	--------	-------------

Ajouter, insérer ou supprimer des règles

Ajouter des règles :

- iptables -A CHAIN REGLE_A_AJOUTER
- iptables [-t TABLE] -A CHAIN REGLE_A_AJOUTER

Insérer des règles :

- iptables -I CHAIN [NUMERO_REGLE] REGLE_A_AJOUTER
- iptables [-t TABLE] -A CHAIN REGLE_A_AJOUTER

Supprimer des règles :

- iptables -D CHAIN REGLE_A_SUPPRIMER
- iptables -D CHAIN NUMERO_REGLE_A_SUPPRIMER

Supprimer toutes les règles d'une chaîne

Vous pouvez supprimer toutes les règles d'une chaîne grâce à l'option « flush », en utilisant la commande :

- `iptables [-t table] -F [chaîne]`

En savoir plus sur
les règles

Les options des règles

Nous avons vu précédemment comment ajouter, insérer ou supprimer des règles. Voici les options les plus courantes à utiliser :

Spécifier l'adresse IP source, le réseau ou le nom de l'équipement :

- `-s` SOURCE
 - `-s 192.168.1.2`
 - `-s 192.168.1.0/24`
 - `-s 192.168.1.0/255.255.255.0`

Spécifier l'adresse IP destination, le réseau ou le nom de l'équipement :

- `-d` DESTINATION
 - `-d 192.168.1.2`
 - `-d 192.168.1.0/24`
 - `-d 192.168.1.0/255.255.255.0`

Les options des règles (2)

Spécifier le protocole avec lequel doit matcher la règle :

- `-p PROTOCOLE`
 - `-p tcp`
 - `-p udp`
 - `-p icmp`

Pour activer un module de correspondance plus étendue :

- `-m MODULE OPTIONS_DU_MODULE`

Les options des règles (3)

Spécifier le port destination :

- `-p PROTOCOLE --dport PORT`
 - `-p tcp --dport 80`
 - `-p udp --dport 53`

Spécifier le port source :

- `-p PROTOCOLE --sport PORT`
 - `-p tcp --sport 80`
 - `-p tcp --sport 22`

Spécifier le type de paquet ICMP :

- `-p icmp --icmp-type TYPE`
 - `-p icmp --icmp-type echo-reply`
 - `-p icmp --icmp-type echo-request`

Les options des règles (4)

Spécifier un target

- -j TARGET/CHAINE
 - -j ACCEPT
 - -j DROP
 - -j MACHAINEPERSONO

Exemples de configuration des règles

Exemple 1

Dans ce premier exemple, nous souhaitons bloquer les paquets qui proviennent de l'IP 192.168.1.10 :

```
iptables -A INPUT -s 192.168.1.10 -j DROP
```

Exemple 2

Dans ce deuxième exemple, nous souhaitons :

- Accepter les paquets provenant du réseau 192.168.1.0/24 pour des demandes de connexion SSH
- Rejeter tous les autres paquets qui essaient d'établir une connexion SSH sans appartenir au réseau 192.168.1.0/24

```
iptables -A INPUT -s 192.168.1.0/24 -p tcp --dport 22 -j ACCEPT
```

```
iptables -A INPUT -p tcp --dport 22 -j DROP
```

Créer et supprimer une chaîne

Pour créer une Chaîne, on utilise la commande :

- `iptables [-t TABLE] -N CHAINE`

Pour supprimer une Chaîne, on utilise la commande :

- `iptables [-t TABLE] -X CHAINE`

Sauvegarder les règles

Les commandes iptables permettent de créer des règles en temps réel, mais au redémarrage de l'équipement, ces règles sont supprimées.

Sur Debian et Ubuntu , on a besoin d'installer le paquet **iptables-persistent**:

- apt-get install iptables-persistent

On utilise la commande suivante :

- netfilter-persistent save