TP Réseau : Pare-feu

Les TP d'acquisition des fondamentaux visent à construire un socle de connaissances de base, à appréhender des concepts, des notions et des modèles qui sont fondamentaux. Ce sont des étapes indispensables pour aborder d'autres apprentissages. Les TP sont conduits de manière fortement guidée pour vous placer le plus souvent dans une situation de découverte et d'apprentissage.

Objectifs

Les objectifs de ce TP sont de découvrir le fonctionnement d'un pare-feu (*firewall*) et d'étudier les règles de filtrage et de translation entre réseaux privées et public.

TP Réseau : Pare-feu

Travail préparatoire

Installation du TP

Le TP5 est disponible dans l'archive `/home/user/sujets-tp/tp5-parefeu.tgz` :

```
host> cd /home/user/
host> tar zxvf sujets-tp/tp5-parefeu.tgz
host> cd /home/user/tp5-parefeu
host> lstart -s
```

La maquette

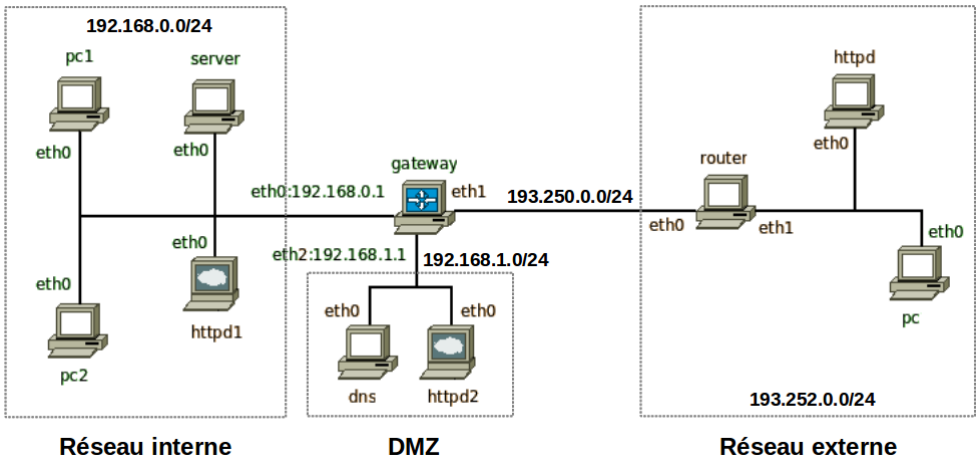
Dans ce TP, la maquette NetKit est la suivante :

```
gateway[0]="A"
gateway[1]="B"
gateway[2]="C"
server[0]="A"
httpd1[0]="A"
router[0]="B"
router[1]="D"
httpd2[0]="C"
dns[0]="C"
httpd[0]="D"

pc1[0]="A"
pc2[0]="A"
pc[0]="D"
```

Le fichier lab.conf

Ce qui donne l'architecture suivante :



Le réseau externe est supposé être le réseau **Internet**. La machine **pc** a été placée dans la maquette pour les tests.

Pare-feu (*firewall*)

Un système pare-feu (*firewall*) est un dispositif conçu pour examiner et éventuellement bloquer les échanges de données entre réseaux.

C'est donc un élément de sécurité d'un réseau qui peut être : un ordinateur, un routeur, un matériel propriétaire. Dans tous les cas, un système pare-feu est une combinaison d'éléments matériels et logiciels.

Le pare-feu joue le rôle de filtre et peut donc intervenir à plusieurs niveaux du modèle OSI.

Il existe trois types de principaux de pare-feu :

- filtrage de paquets (*firewall*)
- filtrage de paquets avec état (*firewall stateful*)
- *proxy*

Le filtrage de paquets est généralement réalisé par des routeurs qui permettent d'accorder ou de refuser l'accès en fonctions des éléments suivants :

- l'adresse source et/ou l'adresse destination
- le protocole
- le numéro de port

Un *firewall stateful* inclut toutes les fonctionnalités d'un filtrage de paquet, auxquelles il ajoute la capacité de conserver la trace des sessions et des connexions dans des tables d'état interne. Tout échange de données est soumis à son approbation et adapte son comportement en fonction des états.

Politique de sécurité

On distingue deux approches :

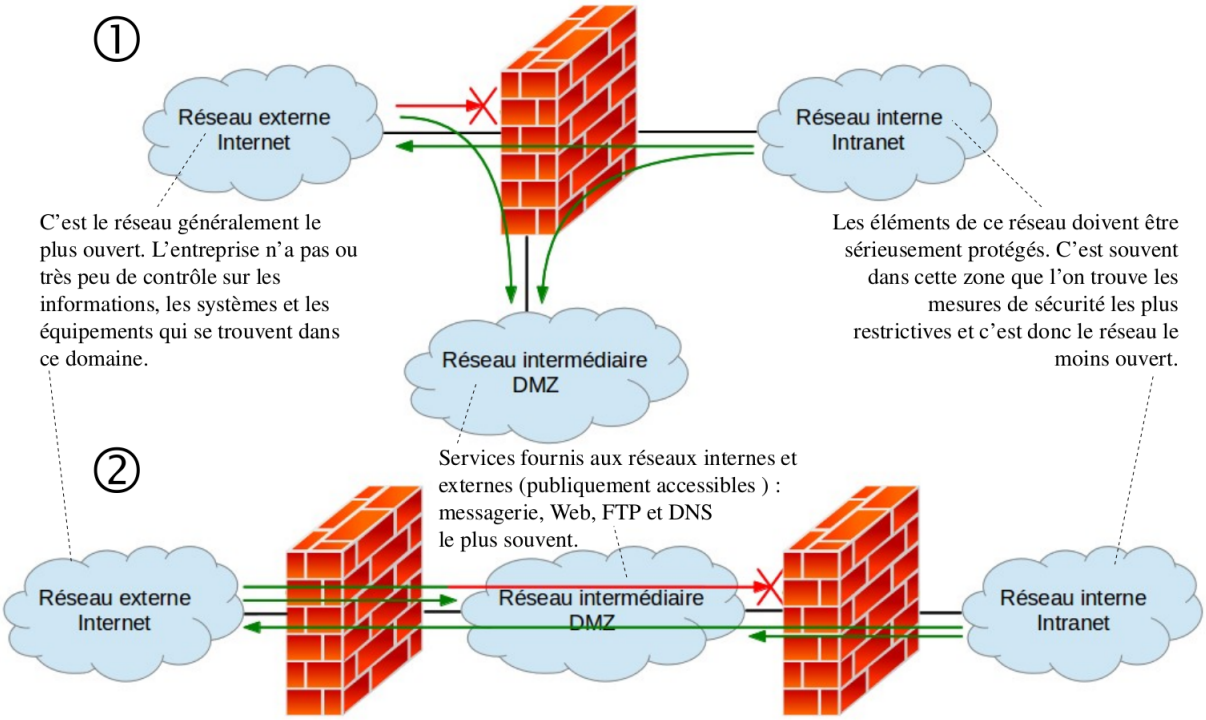
- Politique permissive (*open config*) : cette politique repose sur le principe que par défaut on laisse tout passer puis on va restreindre pas à pas les accès et les services mais la sécurité risque d'avoir des failles.
- Politique stricte (*close config*) : cette politique repose sur le principe inverse. On commence par tout interdire, puis on décide de laisser seulement passer les services ou adresses désirés ou indispensables. La sécurité sera meilleure mais le travail sera plus difficile et cela peut même bloquer plus longtemps que prévu les utilisateurs. C'est évidemment la politique conseillée pour un pare-feu.

DMZ (*DeMilitarized Zone*)

Une DMZ (une zone démilitarisée) est un sous-réseau intermédiaire entre un réseau interne, dit de confiance, et un réseau externe non maîtrisé, donc potentiellement dangereux.

La DMZ isole les machines à accès public (serveurs) du réseau interne. La mise en place d'une DMZ est la première étape de la sécurisation d'un réseau.

On distingue deux types d'architecture :



iptables

iptables est un logiciel libre *GNU/Linux* permettant à l'administrateur système de configurer les règles du pare-feu. Le noyau *GNU/Linux* possède une couche *firewalling* basée sur **Netfilter**. **Netfilter** possède une architecture modulaire.

iptables gère des tables (des tableaux) qui contiennent des « chaînes », elles-mêmes composées d'un ensemble de règles de traitement des paquets.

Chaque table est associé à un type de traitement des paquets :

- `FILTER` : pour les opérations de filtrage de paquets (table par défaut)
- `NAT` (*Network Address Translation*) : pour les opérations de traduction d'adresses
- `MANGLE` : pour modifier les en-têtes des paquets

Les cinq chaînes prédéfinies sont les suivantes :

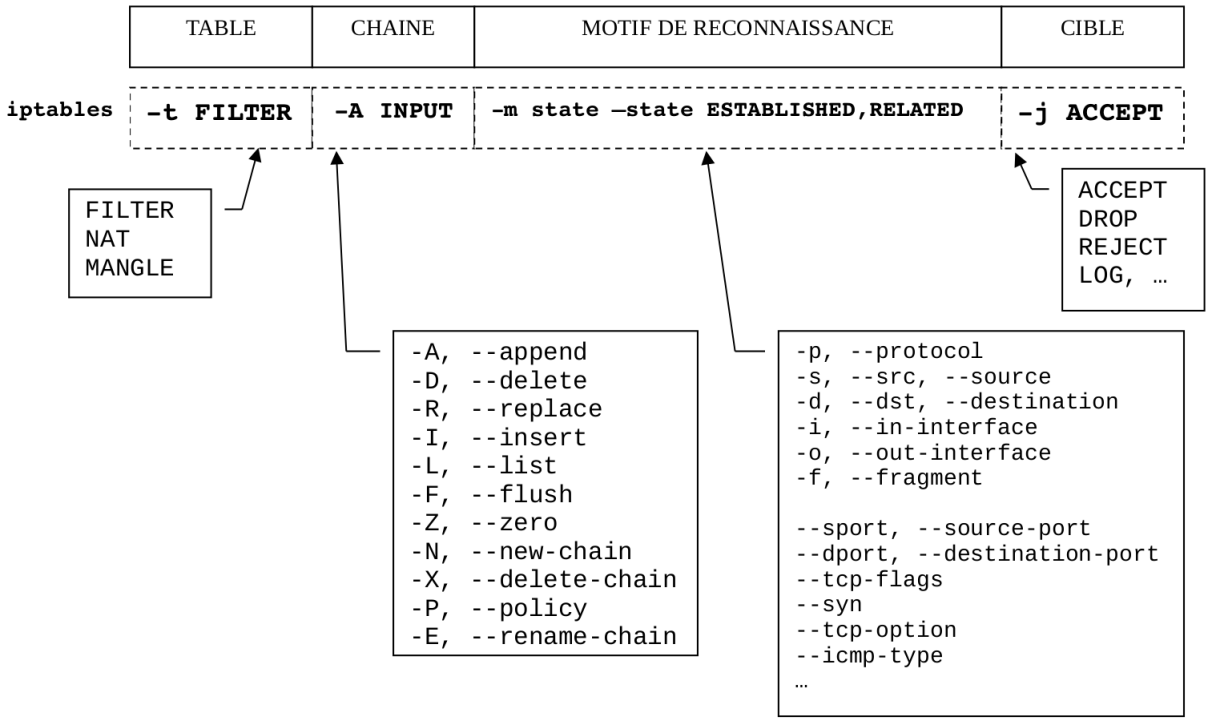
- `PREROUTING` : les paquets avant qu'une décision de routage ne soit prise
- `INPUT` : les paquets à destination de la machine
- `FORWARD` : les paquets traversant (routés par) la machine = ROUTEUR FILTRANT
- `OUTPUT` : les paquets émis par la machine
- `POSTROUTING` : les paquets avant qu'ils ne soient transmis vers le matériel

Les paquets suivent séquentiellement chaque règle des chaînes. Une règle spécifie ce qu'il faut tester dans chaque paquet et ce qu'il faut faire d'un tel paquet. Une règle dans une chaîne peut provoquer un saut à une autre chaîne.

Chaque paquet réseau, entrant ou sortant, traverse donc au moins une chaîne. Tout paquet traité par le système traversera une et une seule chaîne.

Exemple : La table `FILTER`

C'est la table qui permet les opérations de filtrage IP. Les paquets y sont acceptés (`ACCEPT`), refusés (`DROP` ou `REJECT` avec renvoi d'un paquet erreur), logués (`LOG`) ou encore mis en queue (`QUEUE`), mais jamais modifiés. Cette table contient trois chaînes de base : `INPUT`, `FORWARD` et `OUTPUT`.



Lorsque la table n'est pas précisée, **iptables** opère sur la table `FILTER` :

```
# On interdit tout ce qui sort de l'interface loopback
iptables -A OUTPUT -o lo -j DROP

# On interdit tout ce qui rentre du réseau
iptables -A INPUT -s 0/0 -j DROP

# On interdit toutes les requêtes echo (ping)
iptables -A INPUT -p icmp --icmp-type 8 -j DROP

# On interdit toutes les réponses echo (ping)
iptables -A INPUT -p icmp --icmp-type 0 -j DROP

# On interdit l'accès au serveur web du réseau 192.168.1.0
iptables -A INPUT -p TCP -d 192.168.1.0/24 --dport http -j DROP
```

Une initialisation d'**iptables** pour une politique *close config* :

```
# Choix de la configuration par défaut :
# DROP -> close config
# ACCEPT -> open config
POLICY="DROP"

# politique par défaut pour FILTER
iptables -P INPUT $POLICY
iptables -P FORWARD $POLICY
iptables -P OUTPUT $POLICY

# on vide (flush) toutes les règles existantes
iptables -F
iptables -t nat -F
iptables -t mangle -F
```

```
# on efface toutes les chaînes
iptables -X
iptables -t nat -X
iptables -t mangle -X
```

Quelques commandes utiles :

```
// Voir l'ensemble des règles et leur numéro :

pc:~# iptables -L -v --line-number

// Vider toute la table de filtrage :

pc:~# iptables -F

// Effacer la règle numéro 3 de la chaîne INPUT :

pc:~# iptables -D INPUT 3
```

La table « nat » est utilisée pour les translations d'adresses. Comme pour le filtrage, cette table utilise trois chaînes :

- `PREROUTING` pour faire du **DNAT** (*Destination NAT*), la translation est réalisée sur l'adresse de destination avant le processus de routage. Exemple : pour un paquet entrant vers un serveur web interne et masqué, le routeur va remplacer sa propre adresse IP par l'adresse du serveur web.
- `POSTROUTING` utilisée à la sortie du routeur pour faire du **SNAT** (*Source NAT*), l'adresse source est masquée après le processus de routage. Exemple : un ordinateur local veut sortir sur Internet, le routeur va remplacer l'adresse IP du paquet émis en local par sa propre adresse.
- `OUTPUT` pour les paquets qui sont générés localement et qui sortent d'une interface.

Pour les deux premières chaînes, le traitement permettant de faire du masquage est noté **MASQUERADE**.

Cette commande ajoute une règle dans la table de translation d'adresses NAT du routeur qui opère après la décision de routage (*postrouting*) et qui masque (*masquerade*) le trafic provenant du réseau `10.2.0.0` et à destination du réseau `10.3.0.0`. Ce dernier voit le trafic sortant de l'interface `eth2` comme provenant uniquement du routeur.

```
iptables -t nat -A POSTROUTING -s 10.2.0.0/16 -d 10.3.0.0/16 -o eth2 -j MASQUERADE
```

iptables fonctionne aussi en *mode Statefull*. Voici la commande qui autorise tous paquets donc la communication est déjà établie à traverser le *firewall* :

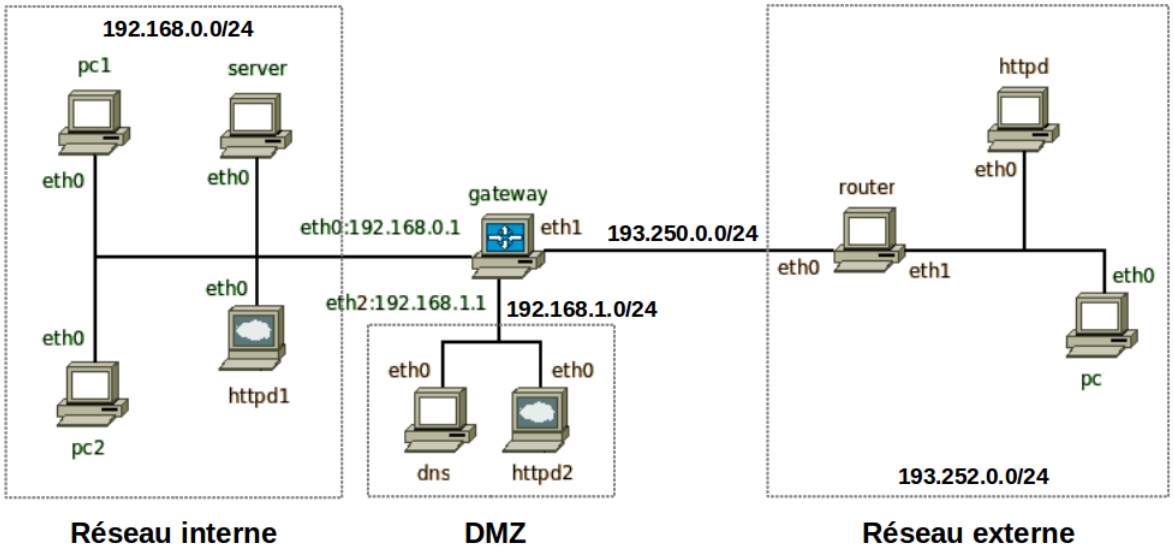
```
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Le paramètre `state` indique l'état de l'échange : demande de connexion (`NEW`) ou établie (`ESTABLISHED` et/ou `RELATED`).

Travail demandé

L'objectif est de configurer un pare-feu (*firewall*) permettant de filtrer les accès vers un réseau interne privé et de rendre accessible à partir d'un réseau externe public (Internet) un serveur web placé sur une zone neutre de type DMZ. Les adresses du réseau interne et de la DMZ seront masquées. Le pare-feu utilisé fonctionne sous *GNU/Linux* avec les règles de filtrage **iptables**.

L'architecture réseau est la suivante :



Vérifier les paramètres IP des interfaces et la connectivité de l'ensemble des machines.

La machine **server** possède un serveur DHCP et DNS pour le réseau interne `192.168.0.0/24`. La machine **httpd1** est un serveur web de type Intranet. La machine **httpd2** est un serveur web que l'on souhaite rendre accessible depuis l'extérieur. La machine **httpd** est un serveur web de type Internet. La machine **gateway** est un routeur pare-feu. **pc** est une machine de test placé dans le réseau externe. Tous les services sont démarrés au lancement de la maquette.

Politique stricte (*close config*)

Question 1. Réaliser une configuration *close config* pour le pare-feu **gateway**. Vérifier maintenant à l'aide de *pings* que les machines (Réseau interne, Internet et DMZ) ne peuvent plus communiquer.

Autorisation des pings

Le but de cette partie est de rajouter les règles sur le *firewall* pour autoriser les *pings* du LAN vers la DMZ.

Question 2. Ajouter les règles de filtrage permettant d'accepter sur l'interface du *firewall* côté LAN un *ping* venant du LAN. Donnez les règles ajoutées. Faire un *ping* à partir du LAN vers l'interface `192.168.0.1` et vérifier que cela fonctionne.

Question 3. Ajouter les règles sur le *firewall* pour autoriser les *pings* du LAN vers les machines situées dans la DMZ et tester.

Question 4. Peut-on pinger l'interface `192.168.1.1` à partir du LAN ? Si non, quelle(s) règles faudrait-il ajouter ?

Filtrage entre le réseau interne et Internet

Les machines du LAN ne doivent pas être directement visibles de l'Internet. On réalisera donc une translation d'adresse pour toute machine ayant comme adresse source le réseau interne et comme réseau

destination le réseau Internet. On doit permettre le *ping* d'une machine du LAN vers Internet (*echo request*) et accepter en retour la réponse du *ping* (*echo reply*).

Question 5. Ajouter les règles de filtrage pour accepter les *pings* vers Internet. Tester et vérifier la translation d'adresse.

Question 6. Ajouter les règles de filtrage pour accepter un accès vers un serveur web situé sur Internet à partir du LAN. Vérifier l'accès avec `lynx`.

Filtrage entre DMZ et Internet

Le serveur web de la DMZ doit être accessible à partir de la machine Internet mais pas directement avec l'adresse `192.168.1.3`. À partir de la machine Internet, seule une connexion avec come URL : `http://193.250.0.1/` doit fonctionner. Pour cela, il faut mettre en place une translation d'adresse du réseau DMZ vers le réseau Internet et un *forwarding* de port pour que toute connexion HTTP venant d'Internet vers la machine `193.250.0.1` soit redirigée vers la machine `192.168.1.3`, sur le port d'écoute du serveur web de la DMZ.

Question 7. Ajouter les règles de filtrage nécessaires pour réaliser la translation d'adresse et le *forwarding* de port. Vérifier l'accès avec `lynx` à partir de **pc**. Analyser les trames échangées avec *wireshark*.

Filtrage entre le réseau interne et DMZ

On souhaite maintenant permettre l'accès au serveur web de la DMZ pour les machines du LAN.

Question 8. Proposer les règles de filtrage pour la chaîne `FORWARD` permettant une connexion du LAN à destination du serveur web sur son port d'écoute. Tester à l'aide du navigateur `lynx` que l'accès au serveur web fonctionne.