Leçon 13 : Le firewall

13.1 Introduction

Le firewall sous Linux est un élément réseau important. Nous l'avons déjà un peu utilisé lorsqu'on a dû activer le NAT lors d'une leçon précédente. Le firewall Linux, nommé `iptables`, est un outil puissant et complexe permettant de gérer et protéger le réseau convenablement.

Devant la complexité de ce dernier, la dernière version de CentOS GNU/Linux propose une gestion alternative, nommée `firewalld` (qui se base sur `iptables` pour fonctionner). Ce firewall plus simple est prévu pour protéger un serveur sur internet : tout le trafic sortant est autorisé, le trafic entrant est filtré et aucun trafic « de passage » n'est prévu.

Devant cette limitation (nous ne pourrions utiliser `firewalld` sur notre machine routeur), nous allons dans cette leçon détailler le firewall traditionnel `iptables`.

Pour éviter tout conflit, il est déconseillé d'utiliser conjointement `firewalld` et `iptables`. Sur un poste CentOS 7, cela revient à désactiver `firewalld` (**déjà fait dans les machines virtuelles**) :

```
$ systemctl stop firewalld
$ systemctl disable firewalld
$ yum install iptables-services
$ systemctl enable iptables
$ systemctl enable ip6tables
```

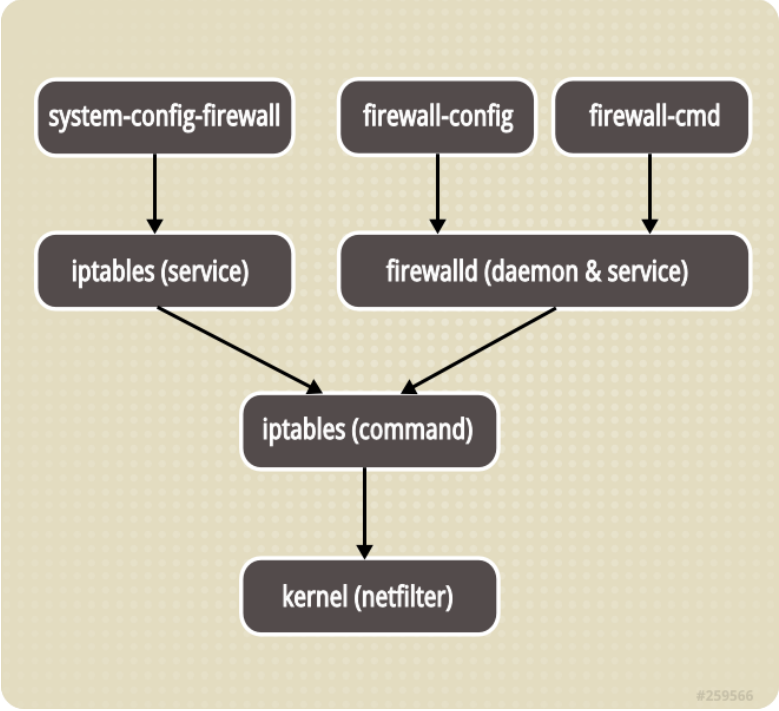

Figure 13.1 : Différences[42] `iptables` et `firewalld`

[42] Source : https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/sec-Using_Firewalls.html

13.2 Aperçu d'IPTables

Sur la figure 13.2, nous pouvons voir schématiquement, le fonctionnement d'`iptables`.

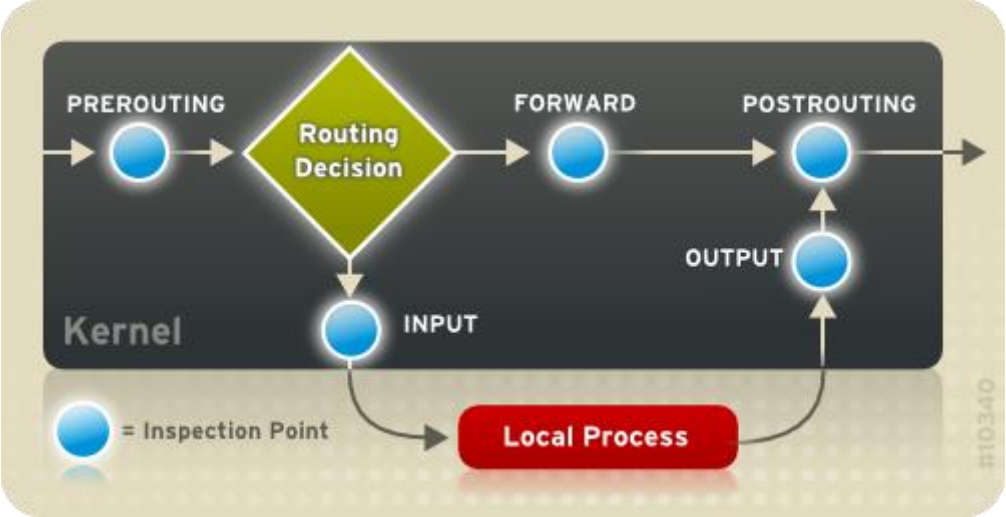

Figure 13.2 : fonctionnement schématique[43] d'IPTables

Lorsqu'un paquet arrive, il passe à travers plusieurs « points d'inspection » que nous appellerons également *chaîne de trafic*.

Tout d'abord, les règles **prerouting** sont appliquées au paquet qui arrive. Ces règles permettent, notamment, de modifier l'adresse IP de destination du paquet. C'est utile lorsqu'il faut « traduire » une adresse IP publique vers une adresse IP privée.

Une fois la phase de **prerouting** passée, le paquet arrive au système de routage. Si le paquet est destiné à la machine locale, il est transmis au point d'inspection (ou chaîne) **input.** Si le paquet est destiné à une autre machine du réseau (dans le cas où la machine est configurée en mode routeur), le paquet est transmis au point d'inspection (ou chaîne) **forward**.

Les règles **input** sont destinées à protéger la machine courante des accès depuis l'extérieur. Ainsi, tout paquet ayant pour destination la machine courante (qu'il vienne du réseau local ou de l'extérieur) est transmis à ce point de décision (ou chaîne). Il est courant dans ces règles d'autoriser uniquement les ports, protocoles et adresses IP autorisées à se connecter à la machine. Une fois autorisé, le trafic est transmis au processus local (programme serveur écoutant sur le port).

Les règles **forward** sont destinées à filtrer les paquets que la machine transmet. Ainsi, ce point de décision (ou chaîne) n'est utilisé que lorsque la machine est configurée en mode routeur. Il est courant dans ces règles d'autoriser les applications disponibles sur le réseau protégé par le routeur. Les règles peuvent aussi bien concerner le trafic émis que le trafic reçu par les machines du réseau.

Sur notre machine routeur, le point d'inspection **input** est utilisé pour tout le trafic reçu, à destination de la machine routeur (dont l'adresse IP destination correspond à l'une des IPs du

[43] Source : https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security_Guide/sect-Security_Guide-IPTables.html

serveur). Le point d'inspection **forward** est utilisé pour tout le trafic à destination ou en provenance de la machine client.

Enfin, le point d'inspection **output** est utilisé pour le trafic qui est émis par la machine courante. Il est courant de placer dans ce point d'inspection (ou chaîne) les règles filtrant le trafic en sortie (bloquer un site, comme facebook.com par exemple, peut facilement être mis en place ici).

Pour terminer, il reste le point d'inspection **postrouting** que nous avons déjà utilisé : les règles prévues ici sont utiles pour modifier le paquet juste avant son expédition. Par exemple, le mécanisme de NAT (ie. la translation d'adresse) se met en place ici.

13.3 Les éléments d'une règle iptables

13.3.1 Le point d'inspection (ou chaîne de trafic)

Le premier élément à déterminer est le point d'inspection dans lequel la règle doit prendre place. Pour les règles de firewall, il y a 3 points d'inspection à considérer : **input, forward** et **output**.

Comme dit précédemment, si le trafic est à destination de la machine locale, c'est dans la chaîne **input** qui faut placer la règle. Si le trafic *traverse* la machine pour atteindre un autre réseau, c'est dans la chaîne **forward** que la règle doit être ajoutée. S'il s'agit du trafic envoyé par la machine locale, c'est dans la chaîne **output** que la règle doit être ajoutée.

13.3.2 La portée

Chaque règle doit être précise pour ne concerner que le trafic souhaité. Ainsi, la portée de chaque règle peut être limitée en spécifiant un ou plusieurs éléments : l'interface réseau d'entrée et/ou de sortie, l'adresse IP source et/ou destination, le protocole réseau ou transport (ICMP, IPv6, TCP, UDP, etc.), le port source et/ou destination (dans le cas de TCP/UDP).

Ainsi, pour cibler le service SSH sur la machine locale, on pourrait ajouter une règle *mentionnant le protocole TCP, à destination dont l'IP est donnée, pour le port destination 22*.

L'ordre des règles dans un point d'inspection (ou d'une chaîne) est important : les règles sont analysées les unes à la suite des autres. Dès qu'une règle correspond au trafic observé, elle est utilisée pour déterminer l'action à entreprendre.

Il est donc important **de commencer par les règles les plus précises** et terminer par les règles plus générales.

13.3.3 L'action

Une fois qu'une règle correspond au trafic observé, il faut déterminer quelle action il faut prendre : *accept* pour mentionner que le trafic est autorisé par le firewall, *reject* pour indiquer que le trafic est refusé (avec une réponse ICMP adéquate), *drop* pour indiquer que le trafic est refusé (sans réponse ICMP, préférable), *log* pour mentionner que le trafic est consigné dans les journaux systèmes.

Il est également possible de préciser un *point d'inspection* comme action de la règle. Cette option est particulièrement intéressante quand l'utilisateur créée ses propres points d'inspection, nous parlerons de cette possibilité plus tard.

13.3.4 Exemples

```
$ iptables -A INPUT -i eno16777736 -d 192.168.190.50 -p tcp --destination-port 22 -j ACCEPT
```

*Cette règle est ajoutée au point d'inspection **input** : le trafic reçu par la machine courante. Elle précise que si le trafic arrive (option `-A INPUT`) par l'interface d'entrée `eno16777736` (option `-i`), à destination de l'adresse IP `192.168.190.50` (option `-d`), dont le protocole de transport est `tcp` (option `-p`), avec comme port destination le port SSH `22` (option `--destination-port`), alors le trafic est autorisé (option `-j ACCEPT`).*

```
$ iptables -A INPUT -d 192.168.131.2 -p udp --destination-port 53 -j ACCEPT
```

*Cette règle est également ajoutée au point d'inspection **input**. Elle précise que le trafic arrivant sur la machine (option `-A INPUT`), à destination de l'adresse IP 192.168.131.2, en utilisant le port UDP (option `-p udp`) DNS 53 (option `--destination-port 53`) est autorisé (option `-j ACCEPT`).*

Dans cet exemple, j'ai choisi de ne pas mentionner l'interface d'entrée (via l'option `-i`). Cela ne pose aucun problème pour autant que la règle soit non-ambigüe. Si je n'avais pas mentionné l'adresse IP destination (option -d), la règle aurait autorisé le trafic à destination du DNS, quelle que soit l'interface réseau qui l'aurait reçu.

13.3.5 Un firewall simple pour un serveur

```
1   #!/usr/bin/perl

2   use strict;

3   my $IPTABLES="/usr/sbin/iptables";
4   my $INTERNAL_NETWORK="10.0.0.0/24";
5   my $DEVICE_NETWORK="eno16777736";
6   my $PORTS_TCP="21,22,80,50000:50500";
7   my $PORTS_UDP="53";

    ### Reset all rules
8   `$IPTABLES -F`;
9   `$IPTABLES -X`;
10  `$IPTABLES -t nat -F`;

    ###
    # INPUT
    ###

11  `$IPTABLES -A INPUT -i lo -j ACCEPT`;
12  `$IPTABLES -A INPUT -i $DEVICE_NETWORK -s $INTERNAL_NETWORK -p icmp -j
    ACCEPT`;
13  `$IPTABLES -A INPUT -i $DEVICE_NETWORK -p tcp -m state --state NEW -m
    multiport --dports $PORTS_TCP -j ACCEPT`;
14  `$IPTABLES -A INPUT -i $DEVICE_NETWORK -p udp -m multiport --dports
    $PORTS_UDP -j ACCEPT`;
15  `$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT`;
16  `$IPTABLES -A INPUT -j LOG`;
17  `$IPTABLES -P INPUT DROP`;

    ###
    # FORWARD
    ###
18  `$IPTABLES -P FORWARD DROP`;
```

```
    ###
    # OUTPUT
    ###
19  `$IPTABLES -P OUTPUT ACCEPT`;
```

Script 13.3 : exemple firewall serveur

Le script 13.3 montre un *exemple de firewall* pour un serveur (qui ne joue pas le rôle de routeur). Nous allons rapidement parcourir les lignes principales pour en expliquer le fonctionnement.

Les lignes `4` à `7` permettent de spécifier la configuration souhaitée : `$INTERNAL_ NETWORK` doit contenir le préfixe du réseau local, `$DEVICE_NETWORK` le nom de l'interface réseau, `$PORTS_TCP` et `$PORTS_UDP`, la liste des ports destination que l'on souhaite ouvrir. Dans notre exemple, on suppose que le réseau local est `10.0.0.0/24`, l'interface réseau est `eno16777736` et que les ports à ouvrir sont les ports TCP 21 (ftp), 22 (ssh), 80 (web), 50000-50500 (tous les ports entre 50000 et 50500). En UDP, le port 53 (dns) doit être ouvert sur la machine.

On suppose donc dans cet exemple que la machine exécute un service Web, FTP, SSH et DNS. De plus, les ports entre 50000 et 50500 doivent également être ouverts.

Les lignes `8` à `10` suppriment toutes les règles qui existeraient au niveau du firewall.

Les lignes `11` à `17` forment le cœur de ce firewall. Il faut bien comprendre que les règles sont analysées séquentiellement et que, lorsqu'une règle *correspond* et *est utilisée*, l'action mentionnée est exécutée et le traitement s'arrête pour ce trafic.

La ligne `11` autorise tout le trafic arrivant par l'interface *loopback*. Il est nécessaire de spécifier cette règle pour éviter un comportement erratique de la machine.

La ligne `12` permet d'autoriser le trafic ICMP (ping, traceroute, …) si la source est le réseau local. Cela permet à la machine de répondre aux requêtes locales alors que les requêtes distantes sont refusées.

La ligne `13` autorise le trafic vers les ports TCP destinations mentionnés dans `$PORTS_TCP`. Le trafic est autorisé uniquement pour les demandes de connexion (`-m state --state NEW`).

La ligne `14` autorise le trafic vers les ports UDP destinations mentionnés dans $PORTS_UDP.

La ligne `15` **est très importante** : elle assure que le trafic qui arrive *qui serait une réponse[44] à une requête envoyée par la machine* est autorisé.

La ligne `16` enregistre dans le fichier journal tout trafic qui n'aurait pas encore été traité par une règle précédente. Le traitement continue ensuite.

La ligne `17` change le comportement par défaut du point d'inspection INPUT. Ainsi, le comportement par défaut est que, tout trafic non traité, est jeté – sans réponse à l'émetteur -.

[44] Si la machine envoie du trafic avec comme port source 1056 et comme port destination 80, la réponse arrivera avec ces ports inversés (port source 80 ; port destination 1056). La règle autorise toute réponse qui émane d'une requête faite par la machine. Sans cette règle, il faudrait écrire une règle spécifique autorisant toute réponse aux requêtes.

La ligne `18` change le comportement par défaut du point d'inspection FORWARD (utilisé pour le mode routeur) en rejetant tout trafic.

La ligne `19`, s'appliquant au point d'inspection OUTPUT (trafic émis par la machine), assure que le comportement par défaut est d'autoriser le trafic. Ainsi, tout paquet émis par la machine est autorisé à sortir.

13.4 Firewall avec mode routeur

Installer un firewall sur une machine jouant le rôle de routeur nécessite une configuration plus précise. Il faut en effet garnir le point d'inspection *forward*. Cependant, en mode routeur, il faut prévoir des règles protégeant la machine routeur et des règles protégeant le réseau derrière celle-ci. Il faut également peut-être *rediriger* du trafic vers cette machine si certaines applications doivent être visibles depuis l'extérieur.

Reprenons notre réseau :

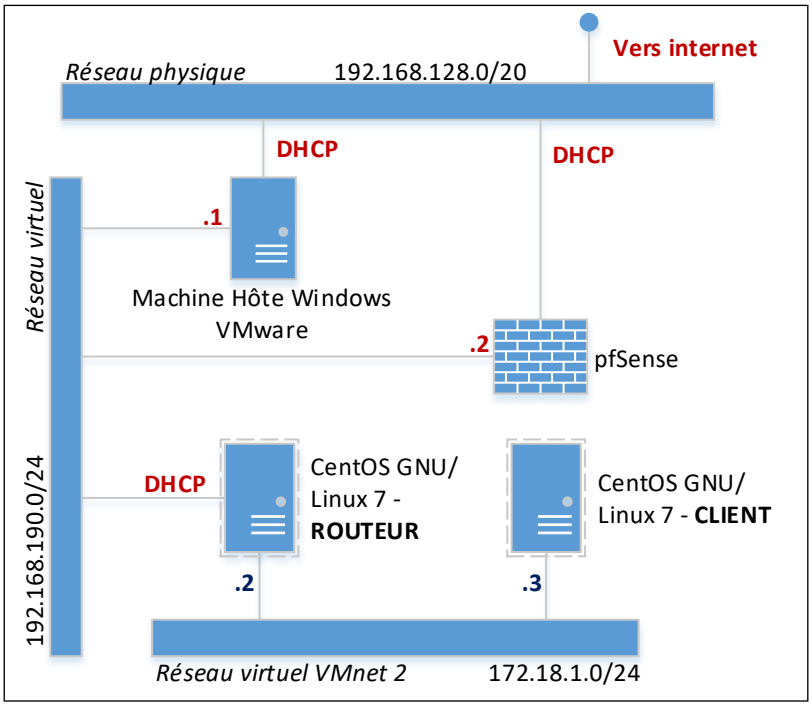

Figure 13.4 : exemple de réseau

Comme nous pouvons voir sur la figure 13.4, la machine routeur dispose de 2 interfaces réseaux et est connectée à la fois aux réseaux 192.168.190.0/24 et 172.18.1.0/24. De plus, la machine routeur réalise une *translation d'adresse* pour le réseau 172.18.1.0/24. Cela signifie que lorsqu'un paquet est transmis par la machine client 172.18.1.3, le firewall change l'adresse IP source par la sienne, 192.168.190.50 (par exemple). Cette translation nous épargne le fait de devoir configurer le réseau 172.18.1.0/24 sur les machines extérieures. En effet, puisque tout le trafic est masqué derrière l'adresse IP de la machine routeur, il faut juste que la machine routeur soit connue, ce qui est le cas.

Imaginons maintenant que nous souhaitons configurer un service FTP ou HTTP sur la machine client et rendre ce dernier accessible depuis l'extérieur, que devons-nous faire ?

Le problème principal vient du fait que le réseau 172.18.1.0/24 n'est pas connu hors de la machine routeur. Donc personne ne sait comment atteindre la machine 172.18.1.3 en dehors de la machine routeur.

Dès lors, comment rendre ce service, actif sur la machine client, accessible ?

Il faut configurer des règles particulières pour indiquer *que le trafic arrivant sur la machine routeur (sur un port déterminé)* ***doit être redirigé vers la machine client***.

```
$ iptables -t nat -A PREROUTING -i eno16777736 -d 192.168.190.50 -p tcp --
destination-port 5022 -j DNAT --to-destination 172.18.1.3:22
```

Dans cet exemple, nous redirigeons le trafic arrivant par l'interface `eno16777736` (`-i eno16777736`), à la machine routeur (`-d 192.168.190.50`) sur port port TCP 5022 (`-p tcp --destination-port 5022`) vers la machine client 172.18.1.3 sur le port 22 (`-j DNAT --to-destination 172.18.1.3:22`).

Ainsi, on peut se connecter au service SSH depuis l'extérieur en se connectant comme suit :

```
$ ssh 192.168.190.50 -p 5022
```

Comme nous le voyons, nous mentionnons l'adresse IP de la machine routeur et le port configuré, celui-ci est redirigé vers la machine client sur le port TCP 22 (ssh).

13.4.1 Créer ses propres points d'inspection

Dans des configurations complexes, comme celle d'une machine linux jouant le rôle de routeur, les règles à prévoir sont souvent nombreuses. Le risque principal réside dans un oubli malencontreux ouvrant la voie à un trafic non désiré.

En effet, dans le point d'inspection INPUT : nous avons le trafic arrivant sur la machine routeur, qu'il provienne de l'extérieur ou du réseau de la machine client. Dans le point d'inspection FORWARD, nous avons le trafic provenant de l'extérieur vers le réseau de la machine client et inversement alors que le point d'inspection OUTPUT reprend le trafic que la machine routeur envoie, aussi bien vers l'extérieur que vers le réseau de la machine client. Comme nous le voyons, les choses ne sont pas simples.

Pour simplifier la configuration de telle configuration, il est intéressant de créer ses propres points d'inspection. Ainsi, dans de tel cas, je conseille de créer des points d'inspection correspondant à chaque direction du trafic. Ainsi, nous pourrions créer les points d'inspection suivants :

- **trust2untrust** : du réseau de la machine client vers l'extérieur
- **fw2untrust** : de la machine routeur vers l'extérieur
- **fw2trust** : de la machine routeur vers le réseau de la machine client
- **untrust2trust** : de l'extérieur vers le réseau de la machine client
- **trust2fw** : du réseau de la machine client vers la machine routeur
- **untrust2fw** : de l'extérieur vers la machine routeur

A l'intérieur de chacun, il est possible de créer des règles, comme celles que nous avons déjà étudiées. Ainsi, nous pouvons autoriser le trafic vers l'extérieur en mentionnant les règles suivantes :

```
$ iptables -A trust2untrust -j ACCEPT
```

```
$ iptables -A fw2untrust -j ACCEPT
```

Nous pouvons ensuite permettre à la machine routeur d'atteindre la machine client :

```
$ iptables -A fw2trust -j ACCEPT
```

On peut, si on le souhaite, permettre au réseau de la machine cliente d'atteindre la machine routeur sans restriction :

```
$ iptables -A trust2fw -j ACCEPT
```

Il reste à configurer les règles destinées à protéger les machines avec le trafic provenant de l'extérieur :

```
$ iptables -A untrust2trust -m state --state RELATED,ESTABLISHED -j ACCEPT
$ iptables -A untrust2trust -p tcp --destination-port 22 -j ACCEPT
$ iptables -A untrust2trust -j LOG
$ iptables -A untrust2trust -j DROP
```

Dans ce premier exemple, nous autorisons uniquement le trafic vers le port 22 (ssh) et les réponses à des requêtes émises par l'une des machines du réseau client.

```
$ iptables -A untrust2fw –m state --state RELATED,ESTABLISHED -j ACCEPT
$ iptables -A untrust2fw -p tcp --destination-port 80 -j ACCEPT
$ iptables -A untrust2fw -j LOG
$ iptables -A untrust2fw -j DROP
```

Dans ce second exemple, nous autorisons uniquement le trafic vers le port 80 (http) et les réponses à des requêtes émises par la machine routeur.

Une fois les règles rédigées, il faut encore utiliser ces nouveaux points d'inspection. En effet, les noms choisis sont sans signification pour le système, il faut donc créer des actions utilisant ces nouveaux points d'inspection comme suit :

```
$ iptables -A INPUT -i lo -j ACCEPT
$ iptables -A INPUT -i eno16777736 -j untrust2fw
$ iptables -A INPUT -i eno33554976 –j trust2fw
$ iptables –A FORWARD –i eno16777736 –o eno33554976 –j untrust2trust
$ iptables –A FORWARD –i eno33554976 –o eno16777736 –j trust2untrust
$ iptables –A OUTPUT –o lo –j ACCEPT
$ iptables –A OUTPUT –o eno16777736 –j fw2untrust
$ iptables –A OUTPUT –o eno33554976 –j fw2trust
```

Les règles mentionnées permettent d'utiliser les points d'inspection créés. Comme nous le voyons, la mention de l'interface d'entrée (option `-i`) et/ou de l'interface de sortie (option `-o`) permet de sélectionner le point d'inspection sans aucune ambiguité.

13.4.2 Script de firewall pour machine routeur

```
1  UNTRUST_IP=192.168.190.50
2  UNTRUST_IF=eno16777736
3  TRUST_NET=172.18.1.0/24
4  TRUST_IF=eno33554976
5  SERVER1_IP=172.18.1.3
6  SERVER1_TCP=21,22,53,80,443,30000:33000
7  SERVER1_UDP=53
```

```
8   iptables -t nat -F
9   iptables -t nat -X
10  iptables -F
11  iptables -X

12  iptables -N untrust2fw
13  iptables -N fw2untrust
14  iptables -N fw2trust
15  iptables -N trust2fw
16  iptables -N untrust2trust
17  iptables -N trust2untrust

    # firewall => trust
18  iptables -A fw2trust -j ACCEPT

    # firewall => untrust
19  iptables -A fw2untrust -j ACCEPT

    # trust => untrust
20  iptables -A trust2untrust -j ACCEPT

    # untrust => trust
21  iptables -A untrust2trust -p tcp -d $SERVER1_IP -m multiport --dports
    $SERVER1_TCP -m state --state NEW -j ACCEPT
22  iptables -A untrust2trust -p udp -d $SERVER1_IP -m multiport --dports
    $SERVER1_UDP -j ACCEPT
23  iptables -A untrust2trust -m state --state ESTABLISHED,RELATED -j
    ACCEPT
24  iptables -A untrust2trust -j LOG
25  iptables -A untrust2trust -j DROP

    # untrust => fw
26  iptables -A untrust2fw -m state --state ESTABLISHED,RELATED -j ACCEPT
27  iptables -A untrust2fw -j LOG
28  iptables -A untrust2fw -j DROP

    # trust => fw
29  iptables -A trust2fw -j ACCEPT

    ## NAT rules
30  iptables -t nat -A PREROUTING -i $UNTRUST_IF -d $UNTRUST_IP -p tcp -m
    multiport --dports $SERVER1_TCP -j DNAT --to-destination $SERVER1_IP
31  iptables -t nat -A PREROUTING -i $UNTRUST_IF -d $UNTRUST_IP -p udp -m
    multiport --dports $SERVER1_UDP -j DNAT --to-destination $SERVER1_IP

32  iptables -t nat -A POSTROUTING -s $TRUST_NET -j MASQUERADE

    ## Distribution rules
33  iptables -A INPUT -i lo -j ACCEPT
34  iptables -A INPUT -i $UNTRUST_IF -j untrust2fw
35  iptables -A INPUT -i $TRUST_IF -j trust2fw
36  iptables -A INPUT -j LOG
37  iptables -A INPUT -j DROP

38  iptables -A FORWARD -i $UNTRUST_IF -o $TRUST_IF -j untrust2trust
39  iptables -A FORWARD -i $TRUST_IF -o $UNTRUST_IF -j trust2untrust
40  iptables -A FORWARD -j LOG
41  iptables -A FORWARD -j DROP

42  iptables -A OUTPUT -o lo -j ACCEPT
43  iptables -A OUTPUT -o $UNTRUST_IF -j fw2untrust
```

```
44 iptables -A OUTPUT -o $TRUST_IF -j fw2trust
45 iptables -A OUTPUT -j LOG
46 iptables -A OUTPUT -j DROP
```

Script 13.5 : firewall pour machine routeur

Le script 13.5 présente une configuration supportant la machine routeur. Il est divisé en 4 sections importantes.

La **première** section (lignes 1 à 7) recense les adresses et interfaces réseaux à prendre en compte : aussi bien du coté *untrust* (interface connectée à internet) que du coté *trust* (interface reliée au réseau interne, celui de la machine client par exemple). Les varibales ainsi définies sont utilisées comme paramètre des commandes de configuration du firewall.

La **seconde** section (lignes 8 à 17) est la phase d'initialisation avec suppression des règles existantes, des points d'inspection avant la recréation de ceux-ci (cela permet d'adapter le script et de le relancer).

La **troisième** section (lignes 18 à 32) configure les règles de firewall dans les points d'inspection créés. Ces règles utilisent les variables configurées à la première section. Les règles nécessaires au fonctionnement du NAT sont également créées (*ouverture* et *redirection* du trafic pour les connexions entrantes et *translation d'adresse* pour les connexions sortantes). Ainsi, les 6 points d'inspection sont configurés (lignes 18 à 29) :

- **fw2trust** : qui est le point d'inspection de *la machine routeur* vers *le réseau interne* est très souvent assez peu surveillé : on autorise ici tout le trafic sans restriction (`-A ACCEPT`)
- **fw2untrust** : qui est le point d'inspection de *la machine routeur* vers le *réseau internet* est également peu surveillé : la machine routeur peut envoyer et se connecter sans restriction à Internet (`-A ACCEPT`)
- **trust2untrust :** qui est le point d'inspection du *réseau interne* vers *le réseau internet* est, dans cet exemple, peu surveillé : le réseau interne peut se connecter n'importe où, avec n'importe quel port vers internet. Il est possible de garnir ce point d'inspection de règles plus restrictives si l'on souhaite empêcher l'accès à certains sites, … au départ du réseau interne.
- **untrust2trust :** qui est le point d'inspection du *réseau internet* vers *le réseau interne* est un des points d'inspection les plus importants à configurer : les règles de protections pour le réseau interne se trouvent ici : on peut autoriser certains serveurs du réseau interne à être accéder depuis l'internet au moyen de règles précises mentionnées ici :
    - c'est le cas de la première règle de ce point qui autorise le trafic TCP vers certaines machines du réseau interne (vers `$SERVER1_IP`, sur les ports `$SERVER1_TCP`). La règle mentionne également qu'il doit s'agir d'une demande d'ouverture de connexion (`-m state --state NEW`).
    - La seconde règle autorise le trafic UDP vers la machine (`$SERVER1_IP`) et les ports (`$SERVER1_UDP`) mentionnés.
    - La troisième règle, quant à elle, autorise le trafic qui serait une réponse à une requête envoyée précédemment (`-m state --state ESTABLISHED,RELATED`). Cette règle doit, dans la plupart des configurations, être présente.
    - Les 2 dernières règles, classiques, enregistrent dans les journaux (`-j LOG`) le trafic inadéquat et jettent celui-ci de manière silencieuse – sans réponse (`-j DROP`).

- **untrust2fw :** qui est l'autre point d'inspection important de la configuration, matérialisant les règles à appliquer *du réseau internet* vers *la machine routeur*. Dans ce point, il s'agit de protéger la machine routeur correctement. Dans notre exemple, il n'y a pas beaucoup de règles proposées :
    - La première règle autorise le trafic qui serait une réponse à une requête envoyée précédemment (`-m state --state ESTABLISHED,RELATED`). Cette règle doit, dans la plupart des configurations, être présente.
    - Les 2 règles suivantes, classiques, enregistrent dans les journaux (`-j LOG`) le trafic inadéquat et jettent celui-ci de manière silencieuse – sans réponse (`-j DROP`).
- **trust2fw :** qui est le point d'inspection du *réseau interne* vers *la machine routeur* est très souvent assez peu surveillé : on autorise ici tout le trafic sans restriction (`-A ACCEPT`).

A la fin de la **troisième** section, les lignes `30` à `32` configurent l'ouverture des ports et le NAT. Ainsi :

- Les lignes `30` et `31` assurent que le trafic entrant (sur l'interface internet `$UNTRUST_IF`, avec l'IP du routeur comme destination `$UNTRUST_IP`), sur les ports mentionnés (en TCP `$SERVER1_TCP` ou en UDP `$SERVER1_UDP`) est bien redirigé vers la machine interne dont l'adresse est configurée (avec l'IP `$SERVER1_IP`).
- La ligne `32` est la ligne qui autorise la translation d'adresses en sortie (`-j MASQUERADE`) pour le réseau interne (`-s $TRUST_NET`).

Enfin, la **quatrième** section (lignes `33` à `46`) installe les règles de distribution (envoi du trafic dans les points d'inspection créés et configurés).

Une fois le firewall configuré et le script exécuté, il convient **de tester celui-ci** pour être sûr que les règles mises en place sont conformes à ce qui est souhaité. Une fois le firewall bien adapté, il **faut sauvegarder** cette configuration pour **qu'elle s'active** automatiquement au démarrage de la machine. Pour ce faire, il faut lancer la commande :

```
$ service iptables save
```

13.5 Exercices

On vous demande de :

1. Configurer un firewall sur votre **machine routeur** de sorte à :
    a. Permettre aux machines du réseau interne (`192.168.131.x`) d'accéder à internet (via le NAT)
    b. Ouvrir les ports suivants sur la machine routeur (point d'inspection `untrust2fw`) :
        i. Autoriser le port SSH (tcp/22) depuis l'extérieur
        ii. Autroriser le port DNS (port udp/53 et tcp/53) depuis l'extérieur
        iii. Autoriser, uniquement depuis les adresses `192.190.190.x`, les requêtes ICMP.
    c. Limiter le trafic sortant provenant du réseau interne (point d'inspection `trust2untrust`) au port web (tcp/80 et tcp/443) et dns (udp/53).
    d. Ouvrir le port `4022` (TCP) et le rediriger vers la machine client. Configurer le port SSH de votre machine client sur ce port `4022`. Votre machine client doit être être accessible, en SSH, via ce port.

2. Configurer sur votre **machine client** un service FTP comme suit :
    a. Le port d'écoute doit être le port TCP `4000` (à la place du port `21`)
    b. Le mode passif doit être configuré sur les ports `63000` à `63500`

3. Configurer un firewall sur votre **machine client** de sorte à :
    a. Permettre l'accès au port SSH `4022`
    b. Permettre l'accès au service FTP configuré au point 2 et aux ports configurés pour le mode passif.

Tester votre configuration depuis la machine routeur (pour le FTP) et depuis l'hôte Windows 7 (pour la configuration SSH).