Chapitre 8 – Routage filtrant (pare-feu Netfilter) – Partie 1

Sommaire

1. Installation du serveur US3.....2
2. Modifications sur DS1.....7
3. Mise en place du pare-feu.....7
3.2. Script et règles de base.....7
3.3. Règles concernant les chaînes INPUT/OUTPUT.....12
.....13
3.4. Règles concernant les flux IP traités par la chaîne FORWARD.....14
3.5. Fin du script et tests.....16

1. Installation du serveur US3

On créer la machine virtuelle US3 avec 3 cartes réseau : enp0s3 → accès par pont ; enp0s8 → réseau interne (DMZ) ; enp0s9 → réseau interne (LAN2).

On procéde à une installation du serveur US3 (Ubuntu Server 24.04 et partitionnement automatique).

On rétablie le compte de l'administrateur

```
sio@us3:~$ sudo -i
[sudo] password for sio:
root@us3:~# passwd
New password:
Retype new password:
passwd: password updated successfully
root@us3:~# _
```

Pour avoir le prompt en couleur on décommente force_color_prompt=yes

```
  GNU nano 6.2                                  /root/.bashrc

# uncomment for a colored prompt, if the terminal has the capability; turned
# off by default to not distract the user: the focus in a terminal window
# should be on the output of commands, not on the prompt
force_color_prompt=yes

if [ -n "$force_color_prompt" ]; then
    if [ -x /usr/bin/tput ] && tput setaf 1 >&/dev/null; then
        # We have color support; assume it's compliant with Ecma-48
        # (ISO/IEC-6429). (Lack of such support is extremely rare, and such
        # a case would tend to support setf rather than setaf.)
        color_prompt=yes
    else
        color_prompt=
    fi
fi

if [ "$color_prompt" = yes ]; then
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
```

On se déconnecte avec logout et on se reconnecte en tant que root

On récupére la dernière liste des paquets disponibles avec apt-get update

On configure les interfaces enp0s3 (172.17.101.219 /16), enp0s8 et enp0s9

```
  GNU nano 6.2                     /etc/netplan/00-installer-config.yaml
# This is the network config written by 'subiquity'
network:
  ethernets:
    enp0s3:
      addresses:  [172.17.101.219/16]
      dhcp4:  no
      gateway4:  172.17.250.3
      nameservers:
           addresses:  [8.8.8.8]
    enp0s8:
      addresses:  [192.168.2.254/24]
      dhcp4:  no
    enp0s9:
      addresses:  [192.168.3.254/24]
      dhcp4:  no
  version:  2
```

On applique la configuration avec la commande netplan apply

```
root@us3:~# netplan apply
```

On vérifie la configuration IP à l'aide de la commande ip a

```
root@us3:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
0
    link/ether 08:00:27:84:e1:0e brd ff:ff:ff:ff:ff:ff
    inet 172.17.101.219/16 brd 172.17.255.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe84:e10e/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
0
    link/ether 08:00:27:8e:d5:95 brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.254/24 brd 192.168.2.255 scope global enp0s8
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe8e:d595/64 scope link
       valid_lft forever preferred_lft forever
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
0
    link/ether 08:00:27:ab:98:19 brd ff:ff:ff:ff:ff:ff
    inet 192.168.3.254/24 brd 192.168.3.255 scope global enp0s9
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feab:9819/64 scope link
       valid_lft forever preferred_lft forever
```

On affiche la table de routage d'US3 à l'aide de la commande ip r

```
root@us3:~# ip r
default via 172.17.250.3 dev enp0s3 proto static
172.17.0.0/16 dev enp0s3 proto kernel scope link src 172.17.101.219
192.168.2.0/24 dev enp0s8 proto kernel scope link src 192.168.2.254
192.168.3.0/24 dev enp0s9 proto kernel scope link src 192.168.3.254
root@us3:~#
```

On modifie le fichier /etc/hosts

```
  GNU nano 6.2                                        /etc/hosts
127.0.0.1       localhost.localdomain   localhost
172.17.101.219  US3.sio-exupery.local   US3


# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

On enleve le # de commentaire à la ligne net.ipv4.ip_forward=1 dans le fichier /etc/sysctl.conf afin que le routage soit mis en place après chaque démarrage de la machine

```
  GNU nano 6.2                                 /etc/sysctl.conf
#
# /etc/sysctl.conf - Configuration file for setting system variables
# See /etc/sysctl.d/ for additional system variables.
# See sysctl.conf (5) for information.
#

#kernel.domainname = example.com

# Uncomment the following to stop low-level messages on console
#kernel.printk = 3 4 1 3

###################################################################
# Functions previously found in netbase
#

# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

On met en place la translation d’adresses

```
root@us3:~# iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
root@us3:~# _
```

On vérifie la bonne prise en compte de la règle par iptables -t nat -L -v

```
root@us3:~# iptables -t nat -L -v
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    1    76 MASQUERADE  all  --  any    enp0s3  anywhere             anywhere
root@us3: # _
```

Afin que la translation d'adresses NAT soit activée à chaque démarrage, on créer un fichier .service dans /etc/systemd/system

```
  GNU nano 6.2                  /etc/systemd/system/iptables-rules.service *
[Unit]
Description=firewall iptables rules

[Service]
Type=oneshot
ExecStart=/root/iptables-rules

[Install]
WantedBy=network-pre.target
```

On créer le script /root/iptables-rules

```
  GNU nano 6.2                                    iptables-rules
#!/bin/bash
/sbin/iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE_
```

On fait en sorte que le fichier /root/iptables-rules soit exécutable avec la commande chmod 755

```
root@us3:~# chmod 755 /root/iptables-rules
root@us3:~# ls -l iptables-rules
-rwxr-xr-x 1 root root 73 Apr  9 15:27 iptables-rules
root@us3:~#
```

On active le nouveau service

```
root@us3:~# systemctl enable iptables-rules.service
Created symlink /etc/systemd/system/network-pre.target.wants/iptables-rules.service → /etc/systemd/s
ystem/iptables-rules.service.
root@us3:~#
```

On relance le système. Le service doit être démarré automatiquement. On vérifie l'existence de la règle NAT incluse dans la chaîne POSTROUTING en affichant la table Nat à l'aide de la commande iptables -t nat –L

```
root@us3:~# iptables -t nat -L -v
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    5   342 MASQUERADE  all  --  any    enp0s3  anywhere             anywhere
```

2. Modifications sur DS1

On modifie le mode d'accès réseau pour la carte 1 (enp0s3) : Réseau interne (LAN2)

On modifie la configuration IP de l'interface réseau enp0s3

```
  GNU nano 8.4                                                /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
address 192.168.3.1
netmask 255.255.255.0
network 192.168.3.0
gateway 192.168.3.254
broadcast 192.168.3.255
```

On vérifie la connectivité avec le serveur ROI 172.17.254.1

```
root@DS1: ~#ping -c2 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=112 time=21.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=112 time=16.9 ms

--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 16.892/19.162/21.433/2.270 ms
root@DS1: ~#_
```

3. Mise en place du pare-feu

3.2. Script et règles de base

On écrie le début du script /root/parefeu.sh

```
  GNU nano 6.2                                  parefeu.sh
#!/bin/bash
# Script pour l'établissement des règles du pare-feu
echo "Script pour le pare-feu"

# Initialisation de quelques variables
echo "--> Initialisation des variables :"

## Interface reliée au réseau externe
WAN=enp0s3
## Interface reliée à la DMZ
DMZ=enp0s8
## Interface reliée au réseau interne
LAN=enp0s9
## Pour réduire la longueur des commandes iptables
IPT="/sbin/iptables"

echo " OK."
```

On enregistre le fichier et on lui donne les droits en exécution avec la commande chmod 755

```
root@us3:~# chmod 755 /root/parefeu.sh
root@us3:~#
root@us3:~# ls -l
total 12
-rwxr-xr-x 1 root root   73 Apr  9 15:27 iptables-rules
-rwxr-xr-x 1 root root  412 Apr 10 07:25 parefeu.sh
```

On prévoie en premier la suppression des règles (option -F) et des chaînes (option -X) des tables Filter, Nat et Mangle ainsi que la remise à zéro des compteurs de paquets dans toutes les chaînes

```
# Nettoyage de toutes les règles existantes
echo "--> Vidage des règles existantes et verouillage :"

## Suppression des règles sur les tables FILTER, NAT et MANGLE
## Option -F pour flush
${IPT} -t filter -F
${IPT} -t nat -F
${IPT} -t mangle -F

## Suppression des chaînes sur les tables FILTER, NAT et MANGLE
## Option -X pour delete-chain
${IPT} -t filter -X
${IPT} -t nat -X
${IPT} -t mangle -X

## RAZ du compteur de paquets
## Option -Z pour zero
${IPT} -Z
```

On crée une fonction utilisateur en BASH, on la nomme TableFILTER elle servira à réinitialiser la politique par défaut (option -P) associée aux chaines de la table Filter

```
## Interface reliée au réseau externe
WAN=enp0s3
## Interface reliée à la DMZ
DMZ=enp0s8
## Interface reliée au réseau interne
LAN=enp0s9
## Pour réduire la longueur des commandes iptables
IPT="/sbin/iptables"

echo " OK."

TableFILTER()
{
${IPT} -t filter -P INPUT $1
${IPT} -t filter -P OUTPUT $1
${IPT} -t filter -P FORWARD $1
}

# Nettoyage de toutes les règles existantes
echo "--> Vidage des règles existantes et verouillage :"
```

On revient à la fin du script et on initialise la table Filter, à l'aide de la fonction TableFILTER, de manière à ce que la politique par défaut associée aux chaines soit d'accepter les paquets

```
# Table FILTER positionnée en accès ouvert sur ses trois chaînes
TableFILTER ACCEPT
```

On ajoute, à ce niveau du script, une demande de poursuite de ce dernier et on prévoie, en conséquence, la possibilité de pouvoir le stopper et de remettre en état le serveur comme s'il n'y avait pas de pare-feu. L'ancienne règle pour le NAT doit être remise en place

```
# Demande de poursuite du script
echo "--> Voulez-vous continuer le script (o/n) ?"
read choix
if [ $choix = 'n' ] ; then
echo " Script interrompu. Vous n'avez plus de pare-feu... "
${IPT} -t nat -A POSTROUTING -o ${WAN} -j MASQUERADE
exit 0
fi
```

La règle consiste dans un premier temps à fermer par défaut toutes les portes au niveau de la table Filter pour ensuite les ouvrir une par une. Par contre, les règles des tables Nat et Mangle sont positionnées par défaut en accès ouvert

```
# Ecritures des règles par défaut

## Table NAT : acceptation sur ses trois chaînes
${IPT} -t nat -P PREROUTING ACCEPT
${IPT} -t nat -P OUTPUT ACCEPT
${IPT} -t nat -P POSTROUTING ACCEPT

## Table MANGLE : acceptation sur ses cinq chaînes
${IPT} -t mangle -P PREROUTING ACCEPT
${IPT} -t mangle -P INPUT ACCEPT
${IPT} -t mangle -P FORWARD ACCEPT
${IPT} -t mangle -P OUTPUT ACCEPT
${IPT} -t mangle -P POSTROUTING ACCEPT

## Table FILTER : refus sur ses trois chaînes
TableFILTER DROP

echo " Pare-feu en fonctionnement, blocage maximum. "
```

A ce stade, on lance le script pour vérifier son fonctionnement

```
root@us3:~# ./parefeu.sh
Script pour le pare-feu
--> Initialisation des variables :
 OK.
--> Vidage des règles existantes et verouillage :
--> Voulez-vous continuer le script (o/n) ?
o
 Pare-feu en fonctionnement, blocage maximum.
root@us3:~#
```

On affiche les tables Filter et Nat. Seules les politiques par défaut des chaînes sont en place

```
root@us3:~# iptables -t filter -L -v
Chain INPUT (policy DROP 74 packets, 11361 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
root@us3:~#
```

```
root@us3:~# iptables -t nat -L -v
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
root@us3:~#
```

On constate que nous ne pouvons plus faire de ping depuis US3 sur l'extérieur sur DS1 ni non plus sur les propres interfaces d'US3 ainsi que sur l'adresse de boucle locale

```
root@us3:~# ping -c1 192.168.3.1
PING 192.168.3.1 (192.168.3.1) 56(84) bytes of data.

--- 192.168.3.1 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

root@us3:~# ping -c1 192.168.3.254
PING 192.168.3.254 (192.168.3.254) 56(84) bytes of data.

--- 192.168.3.254 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

root@us3:~# ping -c1 192.168.2.254
PING 192.168.2.254 (192.168.2.254) 56(84) bytes of data.

--- 192.168.2.254 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

```
root@us3:~# ping -c1 172.17.250.3
PING 172.17.250.3 (172.17.250.3) 56(84) bytes of data.

--- 172.17.250.3 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

root@us3:~# ping -c1 192.168.101.219
PING 192.168.101.219 (192.168.101.219) 56(84) bytes of data.

--- 192.168.101.219 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

root@us3:~# ping -c1 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

On relance le script et on l'interromp en saisissant n après le message de demande de poursuite du script afin de vérifier que le serveur US3 soit bien revenu à son état normal c'est-à-dire sans pare-feu

```
root@us3:~# ./parefeu.sh
Script pour le pare-feu
--> Initialisation des variables :
 OK.
--> Vidage des règles existantes et verouillage :
--> Voulez-vous continuer le script (o/n) ?
n
 Script interrompu. Vous n'avez plus de pare-feu...
```

On vérifie que la chaîne POSTROUTING de la table Nat comporte bien de nouveau la politique MASQUERADE mettant en place la translation d'adresses

```
root@us3:~# iptables -t nat -L -v
Chain PREROUTING (policy ACCEPT 9 packets, 1971 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 MASQUERADE  all  --  any    enp0s3  anywhere             anywhere
root@us3: #
```

On vérifie également que la politique par défaut de chaque chaîne de la table Filter spécifie également de nouveau l'action « ACCEPT »

```
root@us3:~# iptables -t filter -L -v
Chain INPUT (policy ACCEPT 42 packets, 7928 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
root@us3:~#
```

3.3. Règles concernant les chaînes INPUT/OUTPUT

On ajoute les règles suivantes au script

```
# Règles conernant les chaînes INPUT/OUTPUT
# Autorisation de quelques communications

## Autorisation des connexions locales
${IPT} -A INPUT -i lo -p all -j ACCEPT
${IPT} -A OUTPUT -o lo -p all -j ACCEPT

## Connexions avec le réseau interne
${IPT} -A INPUT -i ${LAN} -p all -j ACCEPT
${IPT} -A OUTPUT -o ${LAN} -p all -j ACCEPT

## Connexions avec l'extérieur
${IPT} -A INPUT -i ${WAN} -p all -m state --state ESTABLISHED,RELATED -j ACCEPT
${IPT} -A OUTPUT -o ${WAN} -p all -m state ! --state INVALID -j ACCEPT

echo " Communications locales, internes et externes OK. "
```

Après chaque changement sur le script, on doit le relancer

```
root@us3:~# sh parefeu.sh
Script pour le pare-feu
--> Initialisation des variables :
 OK.
--> Vidage des règles existantes et verouillage :
--> Voulez-vous continuer le script (o/n) ?
o
 Pare-feu en fonctionnement, blocage maximum.
 Communications locales, internes et externes OK.
```

On vérifie la présence des règles dans la table Filter

```
root@us3:~# iptables -t filter -L -v
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  lo     any     anywhere             anywhere
    4  1146 ACCEPT     all  --  enp0s3 any     anywhere             anywhere
    0     0 ACCEPT     all  --  enp0s3 any     anywhere             anywhere             state RELAT
ED,ESTABLISHED

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  any    lo      anywhere             anywhere
    0     0 ACCEPT     all  --  any    enp0s3  anywhere             anywhere
    0     0 ACCEPT     all  --  any    enp0s3  anywhere             anywhere             ! state INV
ALID
```

On vérifie qu'un ping sur 127.0.0.1 fonctionne

```
root@us3:~# ping -c2 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.033 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.029 ms

--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1029ms
rtt min/avg/max/mdev = 0.029/0.031/0.033/0.002 ms
```

On vérifie qu'un ping sur 192.168.3.1 fonctionne

```
root@us3:~# ping -c2 192.168.3.1
PING 192.168.3.1 (192.168.3.1) 56(84) bytes of data.
64 bytes from 192.168.3.1: icmp_seq=1 ttl=64 time=0.203 ms
64 bytes from 192.168.3.1: icmp_seq=2 ttl=64 time=0.215 ms

--- 192.168.3.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1011ms
rtt min/avg/max/mdev = 0.203/0.209/0.215/0.006 ms
root@us3:~#
```

On vérifie qu'un ping sur www.google.fr fonctionne ce qui signifiera que la réponse Echo reply ainsi que celle à la requête DNS auront été obtenues

```
root@us3:~# ping -c2 www.google.fr
PING www.google.fr (142.251.39.99) 56(84) bytes of data.
64 bytes from pnpara-ag-in-f3.1e100.net (142.251.39.99): icmp_seq=1 ttl=112 time=21.0 ms
64 bytes from ams15s48-in-f3.1e100.net (142.251.39.99): icmp_seq=2 ttl=112 time=16.9 ms

--- www.google.fr ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 16.889/18.951/21.014/2.062 ms
root@us3:~#
```

3.4. Règles concernant les flux IP traités par la chaîne FORWARD

On autorise les flux IP pour les demandes DNS, http, HTTPS et FTP de la part du réseau interne. On remarque que pour les paquets DNS, il faut aussi autoriser les paquets encapsulant le protocole UDP

```
# Autorisation des flux IP pour la chaîne FORWARD

## DS1/Clients vers DS2 (DNS, HTTP, FTP)
${IPT} -A FORWARD -i ${LAN} -o ${DMZ} -p tcp --dport 53 -j ACCEPT
${IPT} -A FORWARD -i ${LAN} -o ${DMZ} -p udp --dport 53 -j ACCEPT
${IPT} -A FORWARD -i ${LAN} -o ${DMZ} -p tcp --dport 80 -j ACCEPT
${IPT} -A FORWARD -i ${LAN} -o ${DMZ} -p tcp --dport 443 -j ACCEPT
${IPT} -A FORWARD -i ${LAN} -o ${DMZ} -p tcp --dport 21 -j ACCEPT
```

Seules les réponses aux communications initiées par le réseau interne sont autorisées

```
## DS2 vers DS1/clients
${IPT} -A FORWARD -i ${DMZ} -o ${LAN} -m state --state RELATED,ESTABLISHED -j ACCEPT_
```

Seules les demandes HTTP, HTTPS et FTP sont autorisées (on oublie pas de rajouter les lignes pour le DNS)

```
# DS1/clients vers Internet
${IPT} -A FORWARD -i ${LAN} -o ${WAN} -p tcp --dport 80 -j ACCEPT
${IPT} -A FORWARD -i ${LAN} -o ${WAN} -p tcp --dport 443 -j ACCEPT
${IPT} -A FORWARD -i ${LAN} -o ${WAN} -p tcp --dport 21 -j ACCEPT
${IPT} -A FORWARD -i ${LAN} -o ${WAN} -p tcp --dport 53 -j ACCEPT
${IPT} -A FORWARD -i ${LAN} -o ${WAN} -p udp --dport 53 -j ACCEPT
```

Seules les réponses aux communications initiées par le réseau interne sont autorisées (on oublie pas de rajouter les lignes pour le DNS)

```
## Internet vers DS1/clients
${IPT} -A FORWARD -i ${WAN} -o ${LAN} -p tcp --sport 80 -m state --state RELATED,ESTABLISHED -j ACC
${IPT} -A FORWARD -i ${WAN} -o ${LAN} -p tcp --sport 443 -m state --state RELATED,ESTABLISHED -j AC
${IPT} -A FORWARD -i ${WAN} -o ${LAN} -p tcp --sport 21 -m state --state RELATED,ESTABLISHED -j ACC
${IPT} -A FORWARD -i ${WAN} -o ${LAN} -p tcp --sport 53 -m state --state RELATED,ESTABLISHED -j ACC
${IPT} -A FORWARD -i ${WAN} -o ${LAN} -p udp --sport 53 -m state --state RELATED,ESTABLISHED -j ACC
```

On autorise les flux DNS, HTTP, HTTPS et FTP

```
## DS2 vers Internet
${IPT} -A FORWARD -i ${DMZ} -o ${WAN} -p tcp --dport 53 -j ACCEPT
${IPT} -A FORWARD -i ${DMZ} -o ${WAN} -p udp --dport 53 -j ACCEPT
${IPT} -A FORWARD -i ${DMZ} -o ${WAN} -p tcp --dport 80 -j ACCEPT
${IPT} -A FORWARD -i ${DMZ} -o ${WAN} -p tcp --dport 443 -j ACCEPT
${IPT} -A FORWARD -i ${DMZ} -o ${WAN} -p tcp --dport 21 -j ACCEPT
${IPT} -A FORWARD -i ${DMZ} -o ${WAN} -p tcp --sport 80 -j ACCEPT
${IPT} -A FORWARD -i ${DMZ} -o ${WAN} -p tcp --sport 53 -j ACCEPT
${IPT} -A FORWARD -i ${DMZ} -o ${WAN} -p udp --sport 53 -j ACCEPT
${IPT} -A FORWARD -i ${DMZ} -o ${WAN} -p tcp --sport 443 -j ACCEPT
${IPT} -A FORWARD -i ${DMZ} -o ${WAN} -p tcp --sport 21 -j ACCEPT
```

On autorise les demandes externes de résolution DNS, les requêtes HTTP et HTTPS depuis l'extérieur ainsi que les accès au serveur FTP DS2. On autorise aussi les réponses aux communications initiées par DS2 en tant que client

```
## Internet vers DS2
${IPT} -A FORWARD -i ${WAN} -o ${DMZ} -p tcp --sport 80 -m state --state RELATED,ESTABLISHED -j ACC
${IPT} -A FORWARD -i ${WAN} -o ${DMZ} -p tcp --sport 53 -m state --state RELATED,ESTABLISHED -j ACC
${IPT} -A FORWARD -i ${WAN} -o ${DMZ} -p udp --sport 53 -m state --state RELATED,ESTABLISHED -j ACC
${IPT} -A FORWARD -i ${WAN} -o ${DMZ} -p tcp --sport 443 -m state --state RELATED,ESTABLISHED -j AC
${IPT} -A FORWARD -i ${WAN} -o ${DMZ} -p tcp --sport 21 -m state --state RELATED,ESTABLISHED -j ACC
${IPT} -A FORWARD -i ${WAN} -o ${DMZ} -p tcp --dport 80 -j ACCEPT
${IPT} -A FORWARD -i ${WAN} -o ${DMZ} -p tcp --dport 53 -j ACCEPT
${IPT} -A FORWARD -i ${WAN} -o ${DMZ} -p udp --dport 53 -j ACCEPT
${IPT} -A FORWARD -i ${WAN} -o ${DMZ} -p tcp --dport 443 -j ACCEPT
${IPT} -A FORWARD -i ${WAN} -o ${DMZ} -p tcp --dport 21 -j ACCEPT
```

3.5. Fin du script et tests

On ajoute en fin de script la règle pour l'IP MASQUERADING

```
# NAT
${IPT} -t nat -A POSTROUTING -o ${WAN} -j MASQUERADE
```

On exécute le script parefeu.sh en répondant oui à la question

```
root@us3:~# ./parefeu.sh
Script pour le pare-feu
--> Initialisation des variables :
 OK.
--> Vidage des règles existantes et verouillage :
--> Voulez-vous continuer le script (o/n) ?
o
 Pare-feu en fonctionnement, blocage maximum.
 Communications locales, internes et externes OK.
root@us3:~#
```

On test, depuis le navigateur de DD1, les communications Internet avec requêtes HTTPS et DNS, translation d'adresses et routage