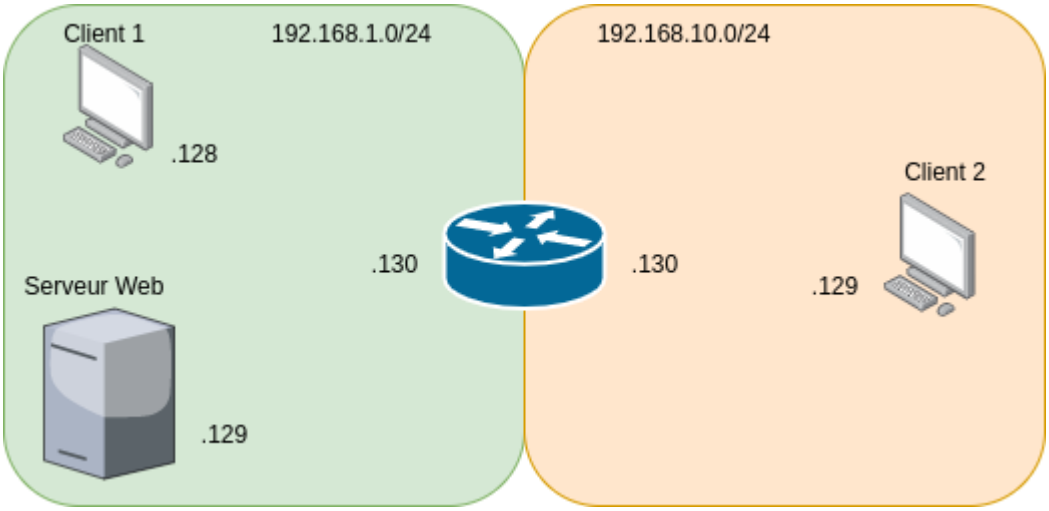
TP4 R401 Alleaume Julien

Routage

Exercice 1 :



Le hook snat ce place sur le POSTROUTING.

Exercice 2 :

```
nft add table Filtreipv4
nft add chain MonFiltreIPv4 apresroute { type nat hook postrouting priority
0 \; }
nft add rule Filtreipv4 apresroute ip saddr 192.168.1.128/24 oif ens37 snat
10.213.10.130
```

Exercice 3 :

Test avec Wireshark :

```
1045 1460.1336582… 192.168.1.128   10.213.10.129   ICMP   98 Echo (ping) request  id=0x742a, seq=100/25
1046 1460.1338409… 10.213.10.129   192.168.1.128   ICMP   98 Echo (ping) reply    id=0x742a, seq=100/25
1047 1461.1578765… 192.168.1.128   10.213.10.129   ICMP   98 Echo (ping) request  id=0x742a, seq=101/25
1048 1461.1580759… 10.213.10.129   192.168.1.128   ICMP   98 Echo (ping) reply    id=0x742a, seq=101/25
1049 1462.1591418… 10.213.10.130   10.213.10.129   ICMP   98 Echo (ping) request  id=0x742a, seq=102/26
```

On vois bien la traduction de 192.168.1.128 à 10.213.10.130 sur la requetes de ping.

Exercice 4 :

```
nft delete chain Filtreipv4 apresroute

nft add chain Filtreipv4 masque
nft add rule Filtreipv4 masque masquerade
```

Test avec wireshark :

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.213.10.130	10.213.10.129	ICMP	98	Echo (ping) request id=0x742a, seq=658/37
2	0.000270084	10.213.10.129	10.213.10.130	ICMP	98	Echo (ping) reply id=0x742a, seq=658/37
3	1.023976950	10.213.10.130	10.213.10.129	ICMP	98	Echo (ping) request id=0x742a, seq=659/37
4	1.024180683	10.213.10.129	10.213.10.130	ICMP	98	Echo (ping) reply id=0x742a, seq=659/37

Exercice 5 :

DNAT doit être placer sur le hook de prerouting.

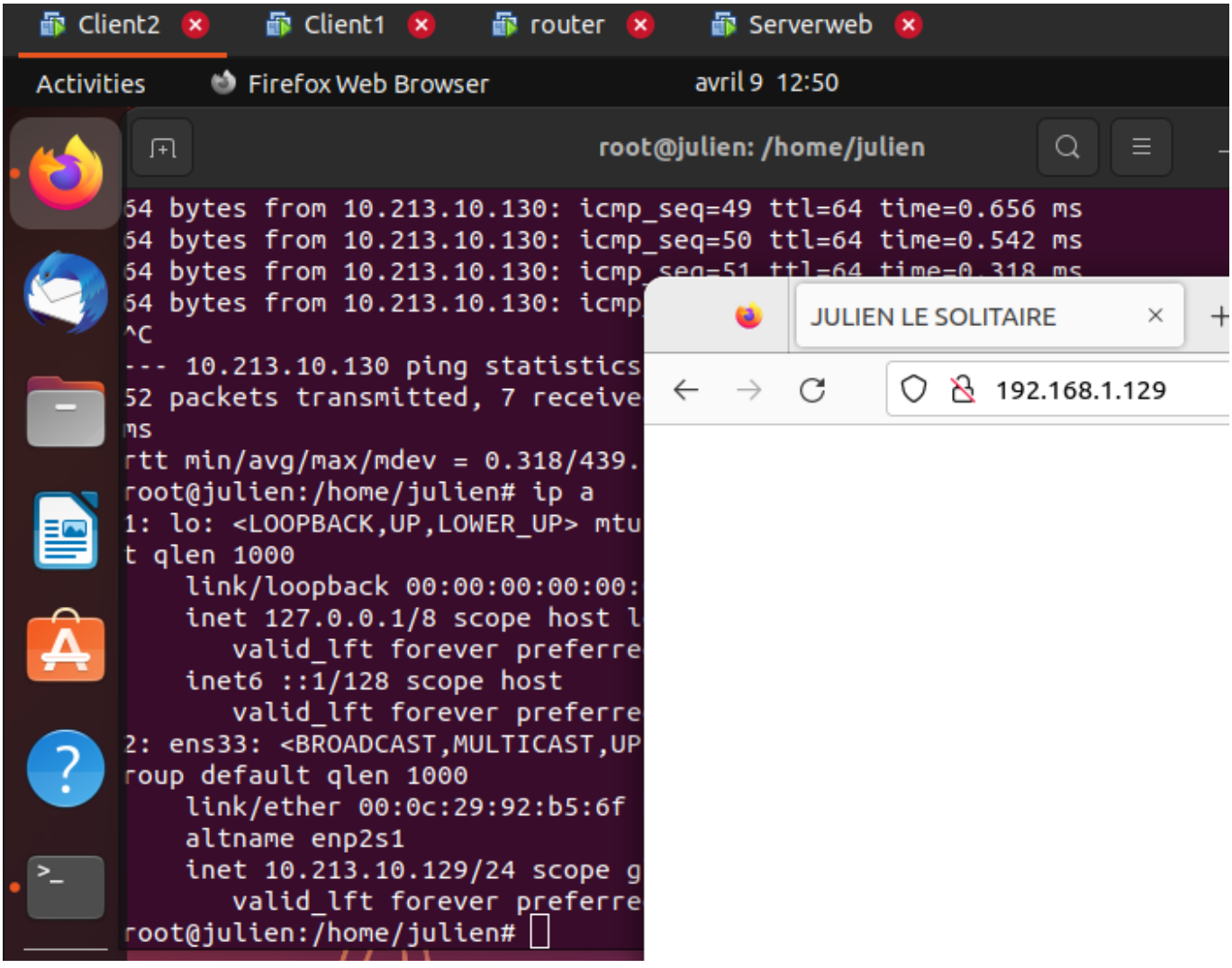
Exercice 6 :

```
nft delete chain Filtreipv4 masque

nft add chain Filtreipv4 dnat e { type nat hook prerouting priority 0 \; }

nft add rule Filtreipv4 distnat iif eth1 tcp dport 8080 dnat 192.168.1.129
```

Exercice 7 :



Le client 2 (10.213.10.129) accéde bien au serveur apache2 192.168.1.129 avec comme nom de site : "JULIEN LE SOLITAIRE".

Exercice 8 :

```
nft add chain ip MonFiltreIPv4 sortie { type filter hook output priority 0
\; }

nft add rule ip filter sortie ip saddr 192.168.1.0/24 tcp dport 80 ip daddr
10.213.10.129 drop
```

Le point négatif de vouloir bloquer des entreprises disposant de beaucoup de trafic est quelle ont beacoup d'adresse publique différentes.

Exercice 9 :

```
nft add chain ip MonFiltreIPv4 ssh { type filter hook output priority 0 \;
}

nft add rule Filtreipv4 ssh ip saddr 10.213.10.0/24 tcp dport 22 drop
```

Cela bloque bien la connexion ssh depuis le réseau externe, la connection ssh tourne en rond avant de déclaré un problème de connexion si on tente de se connecter au routeur.

Exercice 10 :

Le filtre doit être répliquer sur le filtrage exterieur car sinon les info peuvent quand même rentrer sur le réseau mais pas resortir ce qui peut être une faille de sécurité.

La difficulté de filtré un à un les service est que l'erreur est vite arrivé, que ce soit un oublie ou une faute de frappe. De plus il faut mettre a jour souvent la base pour s'adapter au besoins d'une entreprise moyenne par exemple.

On peut utilisé le pricipe du moindre privilèges qui conciste à n'autorisé que le trafic necessaire et de bloquer le reste. Il est possible de faire des groupe de port pour bloquer/autorisé une selection plûtot que un a un.

Exercice 11 :

```
nft add chain ip Filtreipv4 ping {type filter hook input priority 0 \;}

nft add rule Filtreipv4 ping icmp type echo-request drop

nft add rule Filtreipv4 ping icmp type echo-request reject
```

"Drop" de ICMP :

```
PING 10.213.10.130 (10.213.10.130) 56(84) bytes of data.
```

"Reject" de ICMP :

```
From 10.213.10.130 icmp_seq=96 Destination Port Unreachable
^C
--- 10.213.10.130 ping statistics ---
```

Avec fichier de configuration :

```
table ip mon_filtreIPv4 {
        chain input {
                type filter hook input priority filter; policy accept;
                icmp type echo-request reject
        }
}
```

```
table ip mon_filtreIPv4 {
        chain input {
                type filter hook input priority filter; policy accept;
                icmp type echo-request reject
        }
}
```

Exercice 12 :

Création du fichier conf :

```
table inet filter {
    chain input {
        type filter hook input priority 0;

        # Autorisation du trafic ICMP
        icmp type echo-request accept

        # Autoriser SSH depuis le réseau interne
        ip saddr 192.168.1.0/24 tcp dport 22 accept

        # Bloquer tout le trafic entrant
        ct state invalid drop
        ct state {established, related} accept
        drop
    }

    chain forward {
        type filter hook forward priority 0;

        # Autoriser le web
        ip protocol tcp dport 80 daddr 192.168.1.129 accept

        # Bloquer tout le trafic transitant non autorisé
        ct state invalid drop
```

```
        ct state {established, related} accept
        drop
    }
}
```

Exercice 13 :

Je dirai CT pour connection tracking. Cela permet de suivre l'etat des connexion TCP,UDP et ICMP.

Exercice 14 :

```
nft add rule inet filter forward ip saddr 192.168.1.0/24 ip daddr
10.213.10.0/24 ct state established,related accept
```

Je peut bien ping de l'interieur vers l'exterieur mais pas inversement cela est bien bloqué.

Exercice 15 :

```
nft add rule inet filter input ct state invalid counter drop
```

On peut souvent apercevoir cette regles dans les routeur exemple avec la "special dummy rule" dans le pare-feu mikrotik.

Exercice 16-17 :

```
nft add rule inet filter forward ip daddr 192.168.1.129 tcp dport 80 ct
state new conteurs
```

```
nft list conteurs
```

Pour une raison inconnu je n'arrive pas à faire fonctionner cette commande sur mon réseau interne...