TP 4 NAT ET FILTRAGE SOUS LINUX

AKSEL CAUBEL

MATHYS DOMERGUE

RT2 APP IOM

Adressage : PC2 10.213.14.1 Router 10.213.13.1 | 192.168.1.253 PC1 192.168.1.1 Serveur WEB 192.168.1.80

1. LE NAT

1.1. SOURCENAT

EXERCICE 1

Le SNAT doit se mettre sur le hook PostRouting car il s'agit une action de sortie d'interface.

EXERCICE 2

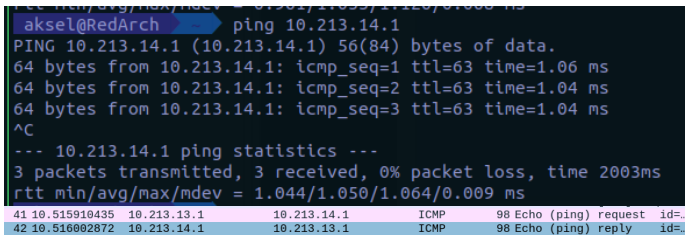
```
nft add table nat

nft add chain nat postrouting { type nat hook postrouting priority 0 \; }

nft add rule nat postrouting ip saddr 192.168.1.0/24 oif eno1 snat 10.213.13.1 // Le "oif" permet dire l'interface de sortie ( Output Interface )
```

EXERCICE 3

Une fois la configuration mise en place, le ping marche. Attention à bien mettre l'adresse DNS si on souhaite avoir la résolution.



1.2 MASQUERADE

EXERCICE 4

```
nft delete rule nat postrouting handle 10 // mon snat étant le handle 10
nft add rule nat postrouting masquerade
```

1.3 DESTINATIONNAT

EXERCICE 5

La règle de DNAT se place sur le hook PreRouting, puisqu'il s'agit d'une règle en entrer du routeur

EXERCICE 6

```
nft add chain nat prerouting { type nat hook prerouting priority 0 \; }  // On ajoute la chaine pour le prerouting
nft add rule ip nat prerouting tcp dport 80 dnat 192.168.1.80:80
```

EXERCICE 7

afin vérifier le bon fonctionnement, il faut que le PC2 (ip : 10.213.14.1) puisse faire accéder au site web depuis l'adresse "public" du routeur :

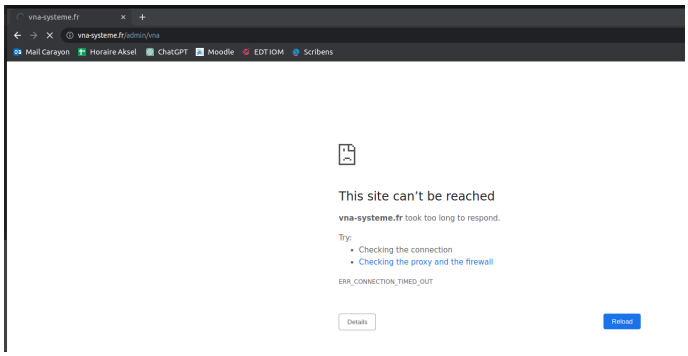
2 FILTRAGE

2.1 SANS ÉTAT : STATELESS

EXERCICE 8

L'adresse IP ne correspondant pas à celle du routeur le paquet va être traité en routage → Forward.

```
nft add rule filter FORWARD ip daddr 141.94.78.218 drop // Bloque l'accès au serveur WEB vna-systeme.fr
```



La limite de cette approche ce n'est que des grosses structures comme YouTube / Facebook ect… C'est qu'ils n'ont jamais qu'un seul serveur. Donc, il faut bannir une grande quantité d'adresse IP qui peuvent être amené à changer. Exemple les services de loadBalancing / les infrastructures K8S qui détruise et recrée des Pods sans arrêt.

EXERCICE 9

Il faut interdire cette fois l'INPUT sur notre routeur pour le protocole TCP sur le port 22 depuis l'interface eno1 (Patte externe du routeur).

```
nft add rule filter INPUT iif eno1 tcp dport 22 reject // Pour le TP j'avertis que je refuse la connexion sinon on fait un drop
```

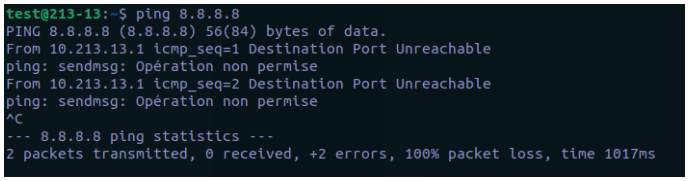
De manière générale, je ne procéderai pas une coupure de tous les accès sur le FireWall et autoriserai uniquement ce dont j'ai besoin.

EXERCICE 10

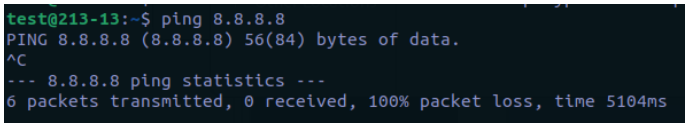
Pour assurer une meilleure sécurité, il faudrait mettre une politique par défaut à "drop" pour que les attaquants ne voient pas les services et de notre côté autoriser un par un les différents services accessibles. Cela évitera ainsi les oublies blocages de port (Le pire des cas sera alors de devoir demander au SI une ouverture de port supplémentaire).

EXERCICE 11

```
nft add rule filter OUTPUT icmp type echo-request reject
```




```
nft add rule filter OUTPUT icmp type echo-request drop
```



EXERCICE 12

Deux possibilités :

- A la fin de la mise en place de nos règles, on vient placer un "drop" qui sera donc exécuté si aucune de nos règles précédentes ne sont pas remplies.
- Lors de l'instanciation de notre chaine, on peut déclarer la "policy" à "drop / reject" selon nos envies (On préfèrera drop pour les raisons évoquées précédemment).

Par habitude, j'utiliserai dans la configuration "drop" à la fin de la configuration (dans le cas d'une policy initial à drop cela rajoutera une sécurité au cas où que l'on est mal fait la première configuration)

Fichier de configuration :

```
nft add rule filter FORWARD icmp type echo-request accept  // Pour ping l'Internet
nft add rule filter FORWARD icmp type echo-reply accept    // Pour ping l'Internet
nft add rule filter INPUT icmp type echo-request accept    // Pour ping le Router
nft add rule filter INPUT icmp type echo-reply accept      // Pour ping le Router
nft add rule filter INPUT tcp dport 22 ip saddr 192.168.1.0/24 accept // accept le SSH du LAN
nft add rule filter FORWARD tcp dport 80 accept // accept les acces WEB http
nft add rule filter FORWARD tcp dport 443 accept // accept les acces WEB https (On va plus loin que la question)
drop
```

2.2 AVEC ÉTAT : STATEFULL

EXERCICE 13

`ct state`

On va pouvoir l'utiliser dans nos règles pour exemple autoriser tout le trafic que l'on a initié :

```
nft add rule filtre forward ct state established accept
```

EXERCICE 14

```
nft add rule filter FORWARD tcp dport 22 daddr 192.168.1.0/24 ct state established accept // Autorise les réponses ssh ( tcp port 22 )

nft add rule filter FORWARD tcp dport 22 saddr 192.168.1.0/24 ct state new accept // Pour qu'il y est established, il faut un new (donc que le LAN puisse initier une connexion)
```

EXERCICE 15

```
nft add counter name cnt_untracked FORWARD ct state untracked // invalide = untracked et on les comptes avec "counter"
```

EXERCICE 16

```
nft add rule filter FORWARD counter tcp dport 80 ct state new accept
```

EXERCICE 17

```
nft list counters
```