Routage de ports

Routage de ports aléatoires:

Le port tcp/5000 local est transféré sur le port 110 d'une autre machine:

```
[root@Poste206 ~]# ssh -fN -L 5000:192.168.1.2:110 root@localhost
root@localhost's password:
[root@Poste206 ~]# telnet localhost 5000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
+OK POP3 mail.domaine.com v2000.70rh server ready
```

Par défault, la connexion au port 5000 ne sera possible qu'à partir de localhost, ce qui limite considérablement l'intérêt de l'opération. En placant la directive « `GatewayPorts` » à « `yes` » dans le ficier de configuration du serveur ssh, la connexion sera possible depuis tout hôte.

Routage de ports

Routage de ports distants:

Le port tcp/5000 de l'hôte distant sera retransmis sur le port 110 d'un autre hôte:

```
[root@Poste206 ~]# ssh -fN -R 5000:192.168.1.1:110 root@192.168.1.5
root@192.168.1.5's password:
[root@Poste206 ~]# ssh root@192.168.1.5
root@192.168.1.5's password:
[root@Poste205 ~]# telnet localhost 5000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
+OK POP3 mail.domaine.com v2000.70rh server ready
```

Attention, dans cet exemple, seule la connexion entre Poste205 et Poste206 est sécurisée par openSSH.

Mode tunnel

OpenSSH permet également de créer un VPN en utilisant les pseudo-devices « `tap` » ou « `tun`». Au lieu de forwarder des ports de niveau 4, il s'agit de proposer une interface virtuelle ethernet ou point-à-point capable de transporter du trafic de niveau 2 ou 3.

Côté serveur, il faudra autoriser cette fonctionalité en positionnant la directive « `PermitTunnel` » à « yes » dans `/etc/ssh/sshd_config`. Le client devra également pouvoir s'authentifier par clé, sans spécifier de mot de passe.

Configuration côté client:

```
[root@Poste206 ~]# cat .ssh/config
Host tunnel
        Hostname Poste205
        Tunnel ethernet
        TunnelDevice 0:0
```

La commande « `ssh -f root@tunnel true`» suffira alors à établir un tunnel entre le client et le serveur.
Il faudra configurer l'interface ethernet virtuelle « `tap0` » sur le client et sur le serveur avant d'utiliser le tunnel.

L'utilisation de TCP au niveau de l'encapsulation n'étant pas idéale, on s'orientera vers des solutions basées sur IPSEC ou openvpn en mode UDP pour créer des VPNs plus robustes.

Sécurisation de l'ouverture du port

Notre objectif est ici de renforcer la sécurité du port TCP ouvert.

ssh étant utileà l'administration distante des machines, il doit être laissé ouvert en permanence, ce qui entraîne une forte exposition aux attaques.

Créer une liste blanche des adresses IP des clients.

```
iptables -A INPUT  -i eth0 -p tcp -s 1.2.3.4 --dport 22 -j ACCEPT
...
```

Limiter le taux de connexion.

Cela permet d'éviter les attaques par force brutale sur le port.

```
iptables -I INPUT -i eth0 -p tcp --dport 22 -m state --state NEW -m recent --set
iptables -I INPUT -i eth0 -p tcp --dport 22 -i eth0 -m state --state NEW -m recent --update \
                  --hitcount 3  --seconds 10  -j DROP
```

Cet exemple limite à trois tentatives de connexion pour dix secondes.

Créer une clé d'entrée

Il faudra envoyer une succession de trames vers des ports TCP determinés et dans un certain ordre pour ouvrir le port 22.

Exemple:

Envoyer une trame vers le port 512 puis une vers le 8356, va ouvrir le port 22 pendant quelques secondes, le temps de se connecter.

Créer une clé d'entrée

```
iptables -N Verif2
iptables -A Verif2 -m recent --name Liste1 --remove
iptables -A Verif2 -m recent --name Liste2 --set

iptables -A INPUT -p tcp --dport 512   -m recent --set    --name Liste1
iptables -A INPUT -p tcp --dport 8356  -m recent --rcheck --name Liste1 -j Verif2

iptables -A INPUT -p tcp --dport 22 -m recent --rcheck --seconds 5 --name Liste2 -j ACCEPT
```

Il suffit ensuite d'envoyer

```
telnet ip_dest 512
telnet ip_dest 8356
ssh ip_dest
```

ou plus efficacement:

```
nmap -r -M1 -PA512,8356 ip_dest
ssh ip_dest
```