TP Réseau

Netcat et autres utilitaires

Olivier LASNE

2020-11-28

TP Réseau

Introduction

Dans ce TP nous allons voir comment utiliser les fonctionnalités réseau de Linux.
À la fois communiquer en TCP, scanner un réseau pour découvrir les machines sur un réseau et les ports ouverts / services disponibles.

Configuration du réseau hôte

Pour ce TP, nous allons utiliser notre machine Kali et Metasploitable.

Créer le réseau privé hôte

Un *réseau privé hôte* est un réseau virtuel, qui connecte des machines virtuelles et qui est accessible *uniquement aux machines virtuelles de VirtualBox*, et à la machine faisant tourner VirtualBox.

Après avoir ouvert **VirtualBox**, cliquer sur **Ficher > Gestionnaire de réseau hôte**. Normalement la configuration suivante est affichée :

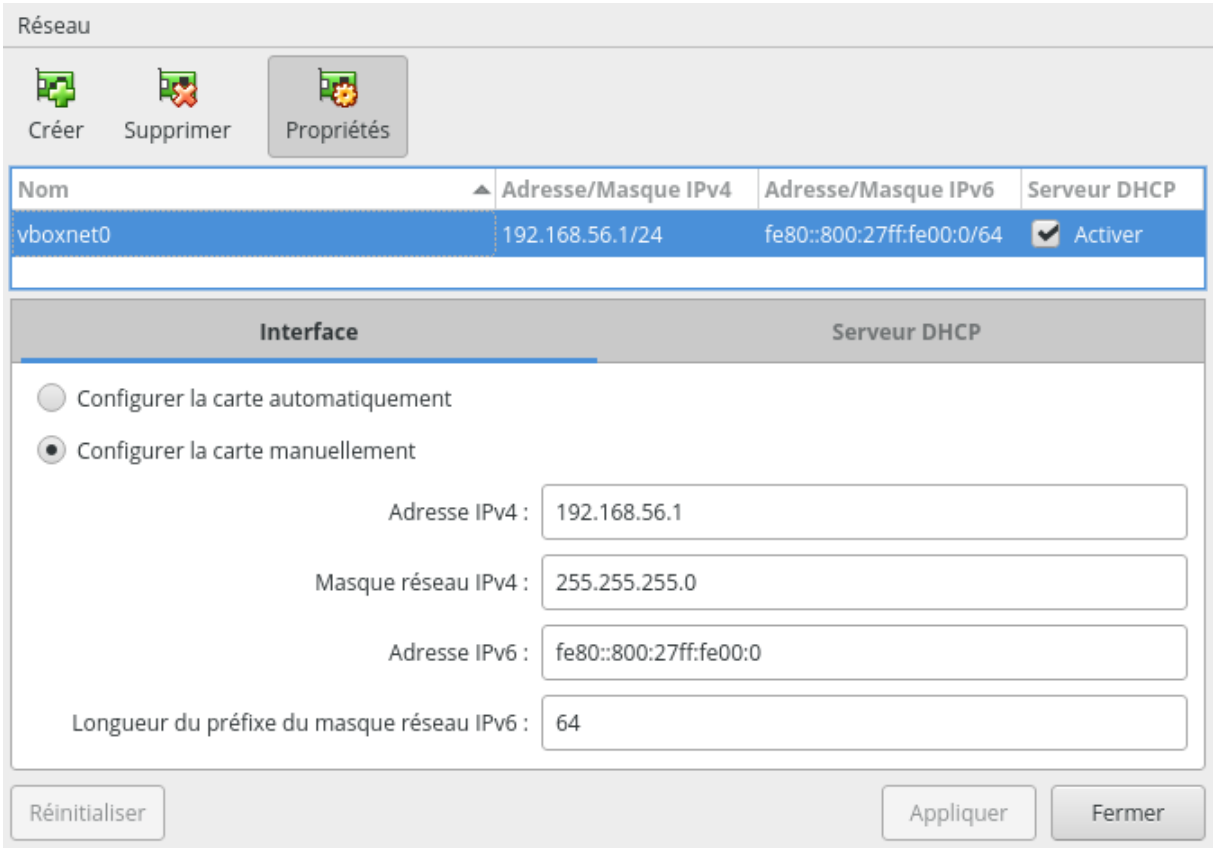

FIG. 1: gestionnaire de réseau hôte

Si il n'y a pas d'interface réseau. Cliquer sur **Créer**, pour créer une nouvelle interface vboxnet0.

Configurer les interfaces de Kali

Pour **Kali** on garde une interface en **NAT** de façon à pouvoir accéder à Internet, et on créer une **seconde interface** pour le **réseau privé hôte**.

1. Éteindre la machine virtuelle Kali.
2. Sélectionner la machine virtuelle dans VirtualBox.
3. Appuyer sur le **bouton configuration** (icone en forme d'engrenage)
4. Dans la barre à gauche cliquer sur **Réseau**
5. Cliquer sur *Interface 2*
6. Cocher *activer l'interface réseau*.
7. Dans **mode d'accès réseau**, selectionner **Réseau privé hôte**.
8. Dans **nom**, selectionner **vboxnet0**

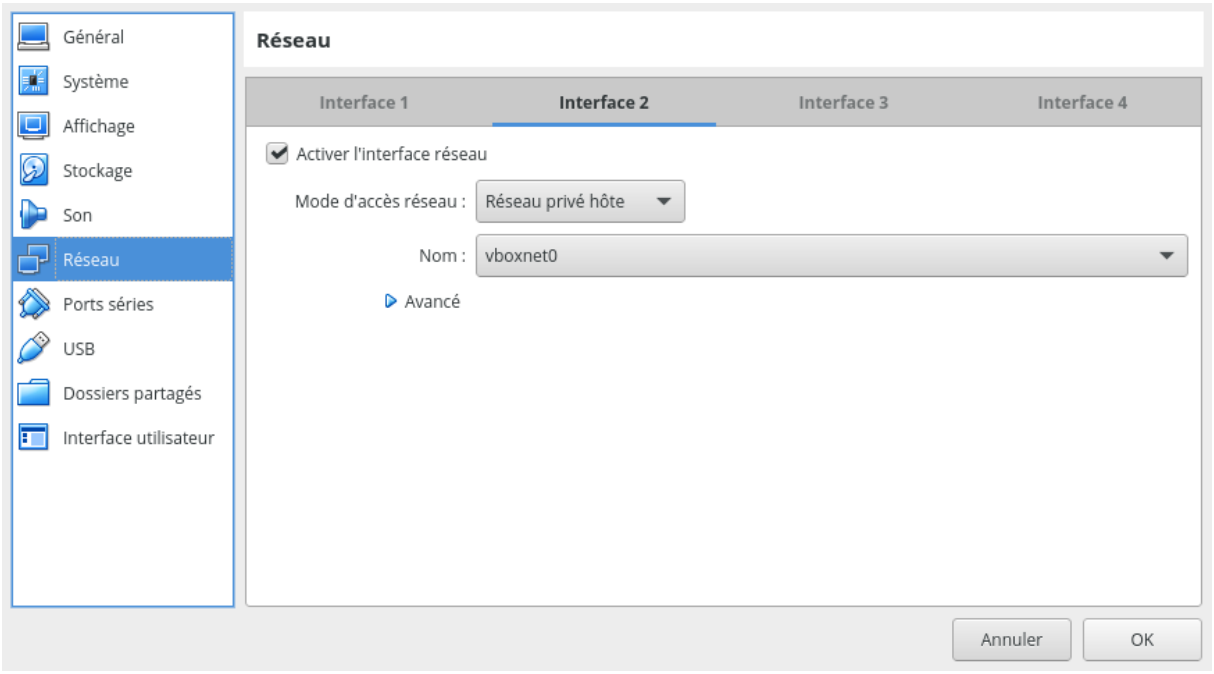

FIG. 2: configuration réseau de Kali

Normalement, nous devrions avoir maintenant deux **interfaces réseau configurées** :

- Une **interface 1** en **NAT**
- Une **interface 2** en **réseau privé hôte**.

Configurer les interfaces de Metaspoiltable

Faire la même chose sur **Metasploitable**.
Configurer l'interface réseau 1, et définir le **réseau privé hôte** *vboxnet0*. (Il n'est pas nécessaire de garder une interface en NAT pour accèder à Internet.)

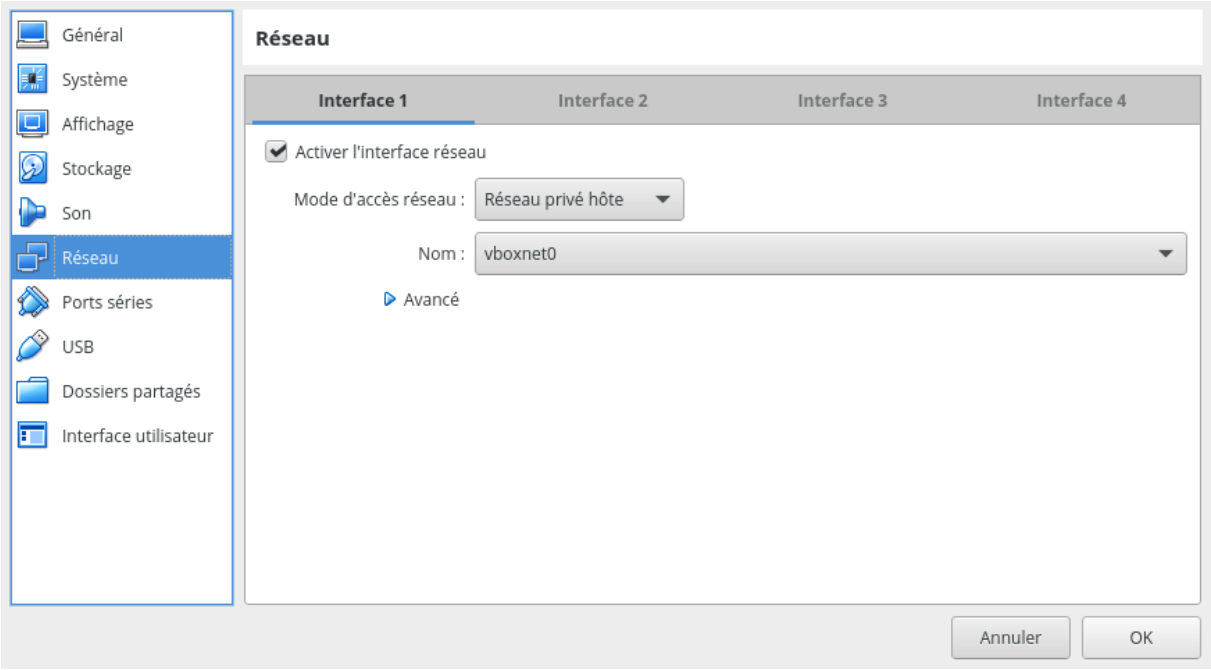

FIG. 3: configuration réseau de Metasploitable

Nous devrions maintenant avoir une seule interface :

- **Interface 1** en **réseau privé hôte**

Commandes réseau sous Linux

Ifconfig

Pour connaitre son adresse **IP**, on peut utiliser la commande **ifconfig**. Sous Windows, la commande est *ipconfig*.

Sur votre machine, la sortie de la commande devrait ressembler à ceci :

```
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fee8:d8bf  prefixlen 64  scopeid 0x20
        ether 08:00:27:e8:d8:bf  txqueuelen 1000  (Ethernet)
        RX packets 29  bytes 3325 (3.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 52  bytes 4655 (4.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.110  netmask 255.255.255.0  broadcast
           192.168.56.255
        inet6 fe80::a00:27ff:fe3b:eec1  prefixlen 64  scopeid 0x20
        ether 08:00:27:3b:ee:c1  txqueuelen 1000  (Ethernet)
        RX packets 44  bytes 6359 (6.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 59  bytes 8642 (8.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Boucle locale)
        RX packets 16  bytes 796 (796.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 16  bytes 796 (796.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Nous avons deux interfaces réseau ici :

- **eth0**
  - Elle correspond à l'interface 1 dans Virtual Box
  - Nous l'avions configuré en **NAT**
  - Elle a pour adresse **IPv4** 10.0.2.15
  - Elle possède une adresse **IPv6** fe80 ::a00 :27ff :fee8 :d8bf
  - Son **adresse matérielle** est 08 :00 :27 :e8 :d8 :bf (paramètre ether)
  - C'est avec elle que nous communiquons à Internet et notre réseau local

- **eth1**
  - Elle correspond à l'interface 2 dans Virtual Box
  - Nous l'avions configuré en **réseau privé hôte**
  - Elle a pour adresse **IPv4** 192.168.56.110
  - Elle possède une adresse **IPv6** fe80 ::a00 :27ff :fe3b :eec1
  - Son **adresse matérielle** est 08 :00 :27 :3b :ee :c1 (paramètre ether)
  - Elle communique uniquement avec les machines de vboxnet0 (ici Metasploitable)
- **lo**
  - Il s'agit de la **boucle locale**
  - Elle est accessible uniquement à notre machine
  - Elle a pour adresse **IPv4** 127.0.0.1
  - Elle possède une adresse **IPv6** ::1

Dhclient

Si une de vos interfaces n'a **pas d'adresse IP**, il est possible d'en demander une au serveur DHCP avec la commande **dhclient**.

Exemple avec eth1 :

```
sudo dhclient eth1
```

Table de routage

Il est possible de regarder votre table de routage IP avec la commande **route -n**.
La commande *route* permet également de modifier cette table.

```
$ route -n
Table de routage IP du noyau
Destination     Passerelle      Genmask           Indic   Iface
0.0.0.0         10.0.2.2        0.0.0.0           UG      eth0
10.0.2.0        0.0.0.0         255.255.255.0     U       eth0
192.168.56.0    0.0.0.0         255.255.255.0     U       eth1
```

La sortie de la commande se lit de la façon suivante :

Pour toutes les IP entre 192.168.56.1 et 192.168.56.255, envoyer les paquets sur l'interface eth1.

Pour toutes les IP entre 10.0.2.1 et 10.0.2.255, envoyer les paquets sur l'interface eth0.

Pour toutes les autres IP, transmettre les paquets à l'IP 10.0.2.2 (passerelle ou gateway) sur l'interface eth0.

Lister les connexions

Il est possible de lister les connexions avec la commande **netstat**.

On va généralement chercher à lister uniquement les ports en écoute avec **netstat -ltupn**.

```
$ netstat -ltupn

Connexions Internet actives (seulement serveurs)
Proto  Adresse locale  Adresse distante  Etat    PID/Program name
tcp    0.0.0.0:22      0.0.0.0:*         LISTEN  817/sshd
tcp6   :::22           :::*              LISTEN  817/sshd
udp    0.0.0.0:68      0.0.0.0:*                 2627/dhclient
```

Ici nous avons le démon (programme qui s'exécute en arrière-plan) ssh sur port 22. Et un client DHCP.

-l : lister seulement les ports **en écoute**
-t : lister les ports **TCP**
-u : lister les ports **UDP**
-n : donner le **numéro** de port, plutôt que le service associé
-p : affiche les **processus** qui utilise le port (nécessite les droits admin)

Le fichier /etc/hosts

Identifiants de *metasploitable 2* : **msfadmin/msfadmin**

Le fichier */etc/hosts permet* d'associer un **nom à une IP** (de la même manière que le protocole DNS).

Par défaut il contient les lignes suivantes :

```
127.0.0.1       localhost
127.0.1.1       kali

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

L'IP et le nom sont séparés par un caractère **Tab**.

Faire la commande **`ping localhost`** est équivalent à **`ping 127.0.0.1`**.

Ce fichier est très pratique pour ne pas avoir à retenir l'IP de machine qui n'ont pas de nom DNS.

Exercice : Ajouter une entrée pour la machine *Metasploitable 2* au fichier **/etc/hosts** de kali.

Les services

Sous Linux il existes plusieurs programmes qui tournent en arrière plan que l'on appelle les services.

Ces derniers sont en chagre de différentes choses comme la configuration réseau (network-manager) ou l'affichage graphique (x11).

Lorsque l'on ajoute des fonctionnalités à notre machine (comme un serveur web, une base de données). Elle est généralement gérée comme un service.

Gérer les services

L'ensemble des services peut être listée avec la commande `service --status-all`.

Pour voir l'état d'un service particulier on peut utiliser la commande `service nomduservice status`.

Par exemple avec SSH :

```
$ sudo service ssh status●
 ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor
        preset: disabled)
     Active: inactive (dead) since Mon 2020-11-30 04:48:10 EST; 3s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 622 ExecStart=/usr/sbin/sshd -D $SSHD_OPTS (code=exited,
       status=0/SUCCESS)
   Main PID: 622 (code=exited, status=0/SUCCESS)
```

Démarrer / éteindre un service

Pour démarrer un service on utilise le paramètre **`start`** :

```
$ sudo service ssh start
```

Pour éteindre un service on utilise le paramètre **`stop`** :

```
$ sudo service ssh stop
```

Configurer un service au démarrage

Pour qu'un service soit lancer automatiquement au démarrage, on utilise un autre utilitaire nommé **systemctl**. Cette commande permet de configurer **systemd** qui gère tous les services sur les distribu-

tions récentes.

Pour qu'un service soit lancé au démarrage :

```
$ sudo systemctl enable ssh
Synchronizing state of ssh.service with SysV service script with /lib/
    systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable ssh
```

Pour qu'un service ne soit plus exécuter au démarrage :

```
$ sudo systemctl disable ssh
Synchronizing state of ssh.service with SysV service script with /lib/
    systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable ssh
Removed /etc/systemd/system/sshd.service.
Removed /etc/systemd/system/multi-user.target.wants/ssh.service.
```

Utilitaires réseau

Pour cette partie nous allons avoir besoin d'une seconde machine. **Démarrez Metasploitable 2**. Les identifiants sont **msfadmin/msfadmin**. Attention, le clavier est en **qwerty**. Une fois connectés, trouvez l'IP de la machine avec `ifconfig`.

Ping

Pour voir si une machine est accessible on peut utiliser la commande **ping**.

```
1 $ ping eff.org
2 PING eff.org (173.239.79.196) 56(84) bytes of data.
3 64 bytes from vm1.eff.org (173.239.79.196): icmp_seq=1 ttl=47 time=144
     ms
4 64 bytes from vm1.eff.org (173.239.79.196): icmp_seq=2 ttl=47 time=144
     ms
```

/ !\ Windows (hors version serveur) est configuré par défaut pour ne pas répondre à ces requêtes.

SSH

Le protocole **SSH** permet d'obtenir un shell sur une machine distante. La syntaxe est `ssh utilisateur@machine`

```
1 root@kali:~$ ssh msfadmin@192.168.56.102
2 msfadmin@192.168.56.102's password:
3
4 msfadmin@metasploitable:~$ whoami
5 msfadmin
```

Pour une première connexion à la machine, *ssh* demande d'authentifier la machine avec une empreinte RSA. Une fois validé, vous pouvez vous authentifier avec un mot de passe.

Pour fermer la connexion, utiliser **Ctrl + D** ou la commande `exit`.

Netcat

Netcat est un peu le couteau suisse du protocole TCP/IP.

La syntaxe est la suivante : `nc ip port`.

On peut par exemple récupérer la bannière (identifiant de version) de SSH sur Metasploitable avec la commande suivante.

```
$ nc 192.168.56.101 22
SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1

Protocol mismatch.
^C
```

Il est possible de créer une connexion en écoute avec l'option **-l**. Par exemple pour écouter sur le **port 4444**.

```
$ nc -lvp 4444
Listening on [0.0.0.0] (family 0, port 4444)
```

Exercice :

1. Ouvrez un port **en écoute** avec netcat sur la machine *Metasploitable*.
2. Connectez-vous dessus depuis *Kali* avec **netcat**.
3. Communiquer entre les deux machines avec la connexion que vous venez de créer. Observez ce qu'il se passe.
4. Utiliser **Wireshark** pendant un échange, et observez le trafic.

Transférer des données avec netcat

Netcat est un utilitaire fiable, et on peut utiliser les **pipes** et redirections pour transferer des fichiers à travers **netcat**.

Prennons en exemple le fichier hosts.

`/etc/hosts`

```
127.0.0.1   localhost
127.0.1.1   metasploitable.localdomain  metasploitable

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
```

On peut tranferer un fichier d'une machine à l'autre de la façon suivante.

Exemple :

Kali

```
nc -lvp 4444 > hostname
```

Metasploitable

```
cat /etc/hostname | nc 192.168.1.101 4444
```

Il est bien entendu nécessaire de modifier les IP et les ports par celle de vos machine.

Exercice :

Les commandes vues ci-dessus fonctionnent sur des fichiers binaires. Utiliser `netcat` pour transférer le programme `/bin/ls` d'une machine à l'autre.

Vous ne pourrez pas exécuter le programme sur la seconde machine car metasploitable est en *32 bits*, et kali en *64 bits*. Plus de détails ici https://askubuntu.com/questions/454253/how-to-run-32-bit-app-in-ubuntu-64-bit.

Néanmoins vous pouvez vérifier que le programme n'a pas été altéré pendant la transmission avec la commande `sha256sum`.

```
$ sha256sum ls
8c0d752022269a8673dc38ef5d548c991bc7913a43fe3f1d4d076052d6a5f0b6  ls
```

Si le résultat est le même avec le fichier original, et le fichier transmis, c'est qu'ils sont parfaitement identiques.