ssh : VPN

- Objet : Créer un tunnel sécurisé entre deux machines grâce à SSH
- Niveau requis :
  débutant, avisé
- Commentaires : *Contexte d'utilisation du sujet du tuto.* Fix Me!
- Débutant, à savoir :
  - Utiliser GNU/Linux en ligne de commande, tout commence là !.
  - Réseaux et VPN
  - Connaitre son IP
  - La commande SSH
- Suivi :
  à-tester
  - Création par Switch le 13/09/2012
  - Testé par <vous> le <date>
- Commentaires sur le forum : Lien vers le forum concernant ce tuto[1)]

Nota : Contributeurs, les Fix Me! sont là pour vous aider, supprimez-les une fois le problème corrigé ou le champ rempli !

Introduction

SSH servant normalement à se connecter à distance sur une machine de manière sécurisée; offre maintenant un mode VPN. Il n'est pas un serveur VPN comme OpenVPN gérant plusieurs connexions, mais permet déjà de faire du VPN en reliant une machine à un réseau distant de manière sécurisée. Dans ce cas, SSH sera plus simple à configurer qu'une solution VPN complète comme OpenVPN. SSH crée une **interface virtuel tun** de chaque côté pour créer le tunnel.

Nous allons voir un exemple ou un pingouin nommé Tux, pourra accéder à tout le réseau local de sa maison à partir de son ordinateur au boulot.

Dans l'exemple, le réseau local de chez Tux est composé d'une trucBox, configurée pour rediriger les connexions SSH entrantes vers l'ordinateur perso de tux dans la maison. Voir section “configuration NAT de la box” plus bas, si besoin

À l'attaque

Connexion bureau à maison

Habituellement, Tux lorsqu'il est au boulot, sait se connecter à son ordinateur de la maison avec ssh tux@maison . Pour aller plus loin et utiliser le mode VPN de ssh, il va:

- se loger en root sur la machine au boulot.
- préciser qu'il veut se connecte sur la machine à la maison en tant que root, aussi.

- ajouter l'option -w 0:0 à la commande ssh.

```
ssh -w 0:0 root@maison
```

L'option -w pour le mode tunnel. 0:0 définissent le numéro des interfaces tun à utiliser. Any peut être utilisé à la place d'un numéro pour utiliser la première interface disponible

Mais une fois logé sur maison, il remarque ce message:

```
channel 0: open failed: administratively prohibited: open failed
```

Et oui, par défaut le serveur ssh n'est pas configuré pour créer des **tunnels**!
Il vérifie quand même avec la commande :

```
ifconfig -a
```

mais ne voit pas d'interface nommée **tun0**
Il modifie[2] donc le **configuration** du **demon ssh** :

```
nano /etc/ssh/sshd_config
```

avec la ligne:

```
PermitTunnel yes
```

Il relance le demon ssh:

```
service ssh restart
```

Quitte et relance la connexion ssh.

Interface tun maison

De nouveau logé en root sur maison plus de message d'alerte. Il utilise :

```
ifconfig -a
```

pour voir toutes les interfaces, mêmes celles non activées.

```
tun0      Link encap:UNSPEC  HWaddr
00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          POINTOPOINT NOARP MULTICAST  MTU:1500  Metric:1
```

Au passage, un petit :

```
lsmod | grep tun
```

montre que le module **tun** à été lancé par **ssh pour créer l'inteface tun0**.

Il est donc logé **root** à la maison, et commence par y **configurer l'interface virtuelle tun0**, puis l'active:

```
ifconfig tun0 172.16.0.1 netmask 255.255.255.252
```

```
ifconfig tun0 up
```

Un petit

```
ifconfig
```

permet de vérifier que **tun0** à bien une **ip** et est en mode **UP**.

Interface tun boulot

En laissant ouverte la première console qui est connectée à la maison, il ouvre une deuxième console sur son poste au travail.
En root il crée l'**interface virtuelle** côté **boulot** donc:

```
ifconfig tun0 172.16.0.2 netmask 255.255.255.252 up
```

Plus besoin d'être root maintenant.

Test de communication des interfaces tun

A partir du boulot: Un **ping** sur l'ip de l'**interface virtuelle** qui se trouve à la **maison**:

```
ping 172.16.0.1
```

Le ping rebondit; les interfaces communiquent bien !

Routage du boulot

En root, il configure une **route** précisant que pour aller sur le réseau **192.168.1.0/24** (réseau local de la maison), les paquets doivent partir du boulot par **tun0**:

```
route add -net 192.168.1.0 netmask 255.255.255.0 gw 172.16.0.1 tun0
```

Il tente un ping sur une machine se trouvant aussi dans le réseau local de la maison: Fix Me!
merci de me proposer une commande plus propre pour créer la route, je nai pas réussi à utiliser la version `ip/bits`

```
ping 192.168.1.55
```

Mais pas de réponse…

Les messages s'arrêtent à son pc de la maison. Il faut activer l'**ip forwarding**, pour que ce dernier **laisse passer** à travers lui les messages destinés **aux autres machines** de la maison (le réseau local 192.168.1.0/24 entier):

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Puis activer le NAT :

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Test du tunnel créé

Du boulot, il tente de pinger la TrucBox de sa maison en utilisant son ip du réseau local de la maison:

```
ping 192.168.1.254
```

Ça ping pour Tux

l'ip de la trucBox peut varier selon le modèle.

Utilisation

Pour passer à la pratique; il va ouvrir son navigateur internet favori, et tenter d'accéder à un service d'un des potes du réseau local de la maison à partir du boulot. Il entre l'adresse de la TrucBox et accède à la page de configuration de cette dernière comme si il avait été **physiquement** dans le **réseau local de la maison** Il peut donc accéder à tous les services des postes de la maison : partages de fichiers FTP, sites web locaux, Wikis persos ou autres.

Compléments

Pour un usage régulier, Tux peut:

- Utiliser l'identification par clés de SSH au lieu des mots de pass à taper à lamain.
- Automatiser la création des interfaces tun et du tunnel etc…
- Il est possible de faire passer ensuite tout le traffic par le tunnel SSH en créant les bonnes routes.
- etc…
- Pour des besoins plus complexes avec plusieurs connexions, il faut se tourner vers des outils dédiés au VPN (comme OpenVPN etc…).

Configuration de la Box

Dans l'exemple, Tux avait déja configuré la Box internet de sa maison pour quelle sache que toute connexion arrivant par le port 22 (port de SSH par défaut), soit redirigé vers l'ip de son ordinateur dans la maison.

Ce fonctionnement est du au fait que son abonnement internet ne lui procure une IP qui doit servire pour plusieurs machines dans la maison. Lorsqu'un message part d'une machine de la maison, passe par la box qui saura donc à quelle machine envoyer les infos lors du retour.

par contre lorsqu'un message initial arrive de l'extérieur, la box, ne sait pas à qui router ce message parmi les différentes machines du réseau local ! Il faut donc configurer la redirection de ports de la Box. Tux, lui s'est connecté à l'interface de configuration de sa trucBox grâce à un navigateur Web en allant sur l'adresse de la box 192.168.1.254. L'adresse peut varier selon les Box évidement. Une fois avoir entré le mot de pass de la Box, a cherché le menu **“Redirection de port”** (dans les “parties réseau local” par exemple). Puis y à ajouté une redirection avec ces règles:

- Port entrant (ou WAN) **22**
- Protocole **TCP**
- IP LAN (machine devant recevoir les messages redirigés) : Il a mis ici l'ip locale de son PC de la maison (ex: 192.168.1.X).
- Port LAN (sur quel port de la machine local on redirige ?): Il a laissé **22** Car n'utilise pas d'autres serveurs ssh chez lui et n'a pas besoin d'utiliser différents ports.

Il a validé les changements, puis attendu un petit instant. Dès lors, il à été possible d'accéder à son pc local depuis l'extérieur en utilisant son IP publique et le port 22 par ssh. Le routeur recevant la demande sait faire suivre les messages à la bonne machine du réseau local.

Extrait des pages de manuel ssh

```
SSH-BASED VIRTUAL PRIVATE NETWORKS
   ssh contains support for Virtual Private Network (VPN) tunnelling using
the tun(4) network pseudo-device, allowing
   two networks to be joined securely.  The sshd_config(5) configuration
option PermitTunnel controls whether the
   server supports this, and at what level (layer 2 or 3 traffic).
   The following example would connect client network 10.0.50.0/24 with
remote network 10.0.99.0/24 using a point-to-
   point connection from 10.1.1.1 to 10.1.1.2, provided that the SSH server
running on the gateway to the remote net-
   work, at 192.168.1.15, allows it.
   On the client:

         # ssh -f -w 0:1 192.168.1.15 true
         # ifconfig tun0 10.1.1.1 10.1.1.2 netmask 255.255.255.252
         # route add 10.0.99.0/24 10.1.1.2
   On the server:

         # ifconfig tun1 10.1.1.2 10.1.1.1 netmask 255.255.255.252
```

```
        # route add 10.0.50.0/24 10.1.1.1
   Client access may be more finely tuned via the /root/.ssh/authorized_keys
file (see below) and the PermitRootLogin
   server option.  The following entry would permit connections on tun(4)
device 1 from user “jane” and on tun device
   2 from user “john”, if PermitRootLogin is set to “forced-commands-only”:
     tunnel="1",command="sh /etc/netstart tun1" ssh-rsa ... jane
     tunnel="2",command="sh /etc/netstart tun2" ssh-rsa ... john
   Since an SSH-based setup entails a fair amount of overhead, it may be
more suited to temporary setups, such as for
   wireless VPNs.  More permanent VPNs are better provided by tools such as
ipsecctl(8) and isakmpd(8).
```

Liens

- Pages de manuel SSH:

```
man ssh
```

- Page basée sur ce : tuto du site PointSlash

1)

N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !

2)

commande d'étition de fichier NANO