TD - Utilisation de ssh

Le but de ce TP est de maitriser les fonctionnalités du protocole `ssh`. La topologie réseau peut être obtenue en lançant le script de démarrage `/net/stockage/aguermou/AR/TP/8/qemunet.sh` en lui fournissant la description de la topologie réseau à l'aide de l'option `-t` ainsi que l'archive contenant la configuration initialte des machines à l'aide de l'option `-a`. Ceci revient à lancer les commandes suivantes :

— au cremi :
```
# cd /net/stockage/aguermou/AR/TP/8/; ./qemunet.sh -x -t topology -a archive_tp8.tgz
```
— à distance :
```
# cd /net/stockage/aguermou/AR/TP/8/; ./qemunet.sh -d tmux -b -t topology -a archive_tp8.tgz
# tmux a
```

Toutes les actions qui vous sont demandées dans la feuille devront être (sauf exception) faites en utilisant un utilisateur autre que `root`. Sur chaque machine il y a utilisateur ayant comme login (resp. mot de passe) le nom de la machine : Par exemple, sur `dt` il y a un utilisateur `dt` ayant comme login (resp. mot de passe) `dt`.

Utilisation basique

1. Pour permettre une authentification sans mot de passe avec `ssh` il faut au préalable positionner la clé publique correspondant au client (`opeth`) sur la machine serveur (`immortal`). Pour ce faire, il faut :

   (a) Générer (sur la machine cliente) le couple clé privée/clé publique se fait via la commande `ssh-keygen`. Ceci a pour effet de créer le fichier `id_rsa` (resp `id_rsa.pub`) qui correspond à la clé privée (resp. publique) dans le dossier `$HOME/.ssh`.

   (b) Copier depuis la machine cliente la clé publique (`.ssh/id_rsa.pub`) sur la machine serveur. Ceci pourra être fait à l'aide de `scp`.

   (c) Concaténer la clé publique copiée au fichier `$HOME/.ssh/authorized_keys`. `$HOME` correspond ici au dossier personnel de l'utilisateur avec lequel on se connecte sur le serveur.

   Remarque : Dans le cas ou une `passphrase` a été utilisée pour protéger la clé privée, cette dernière vous sera demandée à chaque utilisation de la clé. Pour éviter celà, il sera nécessaire d'utiliser un `ssh-agent`.

2. On va essayer maintenant de mettre en place un alias à une connexion ssh pré-définie. Pour ce faire, il suffit d'éditer le fichier `.ssh/config` sur le machine cliente et y ajouter une section ayant la structure suivante :

   ```
   Host example
   Hostname addr_immortal
   User <username>
   ```

   Ainsi il sera possible d'utiliser maintenant `ssh example` au lieu de `ssh <username>@addr_immortal`.

3. Si nous voulons maintenant nous connecter à la machine grave via la machine immortal, il faut ajouter une nouvelle section dont le champs `Hostname` contient l'adresse de grave. Puis, il est nécessaire d'ajouter l'option `ProxyJump` pour dire qu'on veut accéder à grave via immortal :

```
Host example2
Hostname addr_grave
User <username>
ProxyJump example #example correspond au nom de la section d'immortal
```

Ces alias seront aussi utilisable pour la commande `scp`.

Tunnels

Il y a deux façons de créer un tunnel SSH, le transfert de port local et distant. La meilleure façon de les comprendre est de commencer par un exemple, commençons par le transfert de port local.

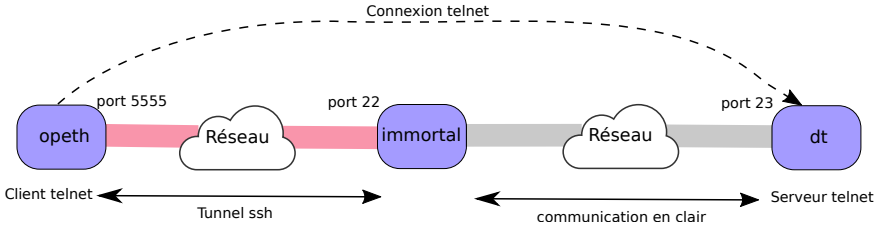

FIGURE 1 – Transfert de port local.

1. Imaginez que vous êtes sur un réseau privé (sur `opeth` par exemple) qui ne permette pas les connexions à un serveur spécifique (`dt` dans notre cas). Pour contourner ce problème, nous pouvons créer un tunnel à travers un serveur (`immortal` dans notre cas) qui n'est pas dans notre réseau et qui peut donc accéder à `dt` (voir Figure 1).

   ```
   ssh -N -f -L 5555:addr_dt:23 <username>@addr_immortal
   ```

   L'option clé est le `-L` qui dit que nous faisons un transfert de port local. Ensuite, il est dit que nous transférons notre port local 5555 vers `dt` sur le port 23, qui est le port par défaut pour telnet. Vous pouvez maintenant vous connecter à `dt` en utilisant le tunnel via la commande suivante :

   ```
   telnet localhost 5555
   ```

   Dans ce contexte, toute connexion locale sur le port 5555 sera transférée automatiquement sur le port 23 de la machine `dt`.
2. Nous allons maintenant nous intéresser au deuxième type de transfert de port : le transfert de port distant. Encore une fois, il est préférable de l'expliquer à l'aide d'un exemple.

   Supposons que vous développez une application web sur votre machine locale (`opeth`), et que vous aimeriez la montrer à un ami. Malheureusement, votre administrateur ne vous a pas fourni d'adresse IP publique, il n'est donc pas possible de se connecter directement à votre machine via Internet.

   Pour résoudre ce problème à l'aide de `ssh`, vous avez besoin d'avoir accès un autre ordinateur (`immortal`), qui est lui accessible depuis internet et qui a un serveur `ssh` (voir Figure 2). Il peut s'agir de n'importe quel serveur sur Internet, à condition que vous puissiez vous y connecter à l'aide de `ssh`. Nous allons donc demander à `ssh` de créer un tunnel qui ouvre un nouveau port sur le serveur, et le connecter à un port local sur votre machine. Ceci peut être fait à l'aide de la commande suivante.

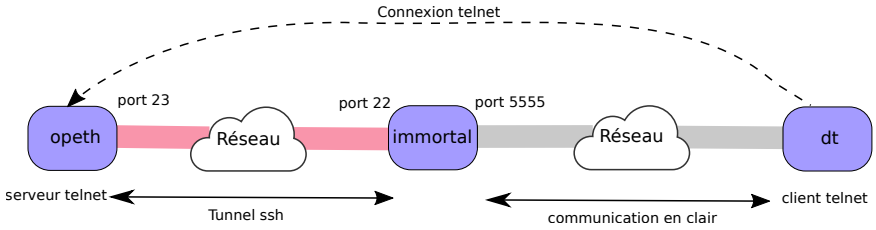

FIGURE 2 – Transfert de port distant.

```
ssh -N -f -R 5555:localhost:23 <username>@addr_immortal
```

La commande peut être interprétée de la manière suivante : Toute connexion sur le port 5555 d'immortal sera redirigée vers le port 23 de la machine locale ce qui rend le serveur écoutant sur le port 23 de la machine locale accessible depuis l'extérieur.

Pour que ceci soit fonctionnel, il faudra au préalable positionner l'option `GatewayPorts` à `yes` dans `/etc/ssh/sshd_config` sur le machine qui sert de pivot (`immortal` dans notre cas) puis redémarrer le démon ssh.

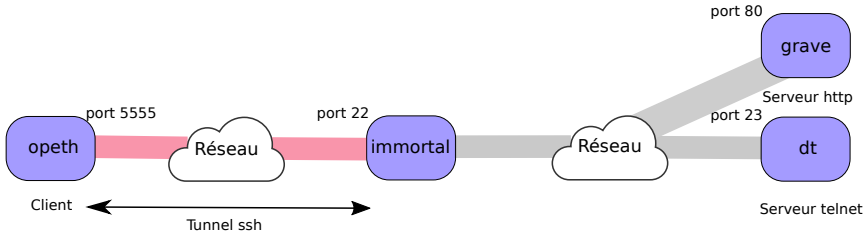

FIGURE 3 – Transfert de port dynamique.

3. Enfin le tunnel le plus avancé est le tunnel dynamique (voir Figure 3). Il sert à transférer via une connexion ssh pré-établie n'importe quelle connexion TCP (contrairement aux cas précédent). La commande pour établir un tunnel dynamique est la suivante :

   ```
   ssh -f -N -D <port> <username>@addr_immortal
   ```

   Le port dans la commande correspond au point d'accès au tunnel sur la machine locale. Ensuite pour utiliser le tunnel de manière dynamique il est nécessaire d'utiliser un outil tel que `proxychains` [1] qui permet de forcer n'importe quelle connexion TCP établie par n'importe quelle application à passer par un proxy `SOCKS` (ssh étant aussi un serveur `SOCKS`). Deux méthodes sont possibles pour faire en sorte que `proxychains` redirige le trafic vers le port d'entrée du tunnel :
   - Modifier le fichier `/etc/proxychains.conf` pour spécifier le port local utilisé à l'étape précédente (i.e. proxychains doit envoyer le traffic sur le port local correspondant au tunnel ssh établi à l'étape précédente). De la documentation relative à cette configuration est fournie sur le site suivante : https://netsec.ws/?p=278. Enfin, Pour faire en sorte qu'une commande se basant sur des connexions tcp utilise le tunnel ssh que vous avez créé, il suffit de la préfixer avec `proxychains`.
   - Soit en positionnant la variable d'environnement `PROXYCHAINS_SOCKS5` à la valeur du numéro de port d'entrée du tunnel avant tout appel a `proxychains`. Par exemple : `PROXYCHAINS_SOCKS5=<port> proxychains <cmd>`.

1. https://github.com/haad/proxychains