Travailler et transférer des fichiers à distance en toute sécurité

L'objectif de ce TP est d'une part de vous présenter un outil permettant de travailler à distance en mode texte ou en mode graphique, en toute sécurité, à savoir l'outil Secure Shell (ou SSH). **IMPORTANT : changer le mot de passe root ! !**

Connexion à distance et transfert de fichiers

1 Solutions non sécurisées à éviter avec un *login / password*

1.1 Se connecter et exécuter des commandes - `telnet`

Il s'agit de savoir ouvrir une session à distance en mode texte. On fait l'hypothèse que la machine locale se nomme *alpha* et la machine distante *beta* (on peut aussi utiliser une adresse IP).

Cet outil permet d'ouvrir une connexion à distance avec une authentification complète par le couple *login / password*. Une fois connecté, vous pouvez travailler sur la machine distante, c'est-à-dire que vous pouvez utiliser des commandes telles que `ls`, `cp`, etc. Il fonctionne selon le principe client / serveur, cela signifie que la machine distante est configurée en serveur, tandis que la machine locale lui demande un service.

— Connexion sur une machine de nom `beta` ou d'adresse IP `172.20.20.15`

```
dupond@alpha~$ telnet beta
dupond@alpha~$ telnet 172.20.20.15
```

— Déconnexion (`exit` ou `logout`)

```
dupond@beta~$ exit
```

Les manipulations à effectuer sont :

1. créer un compte `telnet-user` avec la commande `adduser` en utilisant comme de mot de passe le mot `CqriT` ;
2. installer les parties client et serveur de `telnet`

   ```
   apt-get update
   apt-get install telnet telnetd
   ```

3. installer l'analyseur de réseau `wireshark` si celui-ci n'est pas disponible au niveau du menu `Internet`. La procédure pour installer est la suivante (répondre *<Yes> / <Oui>* à la question `Should non-superusers be able to capture packets?`) :

   ```
   apt-get install wireshark
   usermod -a -G wireshark "$USER"
   ```

En remplaçant `$user` par le login de votre utilisateur, soit en principe `tpreseau`. Ensuite, pour que l'appartenance au groupe `wireshark` soit prise en compte il faudra se déconnecter, puis se reconnecter. Si `wireshark` est déjà installé, commencer par un `dpkg-reconfigure wireshark-common` qui devrait poser la question évoquée ci-dessus, puis faire la manipulation avec la commande `usermod` ;

4. afficher les informations sur l'interface réseau Ethernet via les commandes `ip addr show` et `ip link show`, afin de voir comment est nommée la carte réseau ;
5. lancer `wireshark` et démarrer la capture du trafic réseau en double cliquant sur le nom de l'interface réseau (ou via `Capture`) ;
6. se connecter avec `telnet`, via le compte `telnet-user`, sur la machine d'un de vos camarades, utiliser quelques commandes comme `ls`, puis se déconnecter. Utiliser `man` pour consulter les options de `telnet`. Pour voir qui est connecté sur votre machine, il faut recourir à la commande `who`. Vous pouvez également utiliser la commande `w` ;
7. après avoir mis fin à la connexion, arrêter la capture du trafic réseau, puis étudier les traces affichées par `wireshark`. Pour ce faire, trouver le premier paquet partant de votre machine vers celle de votre camarade (regarder les adresses IP), sélectionner le menu `Analyser`, puis l'item `Suivre` et `Flux TCP`. Que peut-on voir concernant le couple *login / password* utilisé pour se connecter ?
8. finalement désinstaller `telnet` et `telnetd`.

1.2 Transférer des fichiers - `ftp` (*File Transfer Protocol*)

Cet outil permet à des machines dont le système d'exploitation est différent de transférer des fichiers sur un réseau de type TCP/IP. Il fonctionne également selon le principe client / serveur.

— Connexion à un serveur de nom `beta`

```
dupond@alpha~$ ftp beta
```

À la place d'un nom de machine on peut également spécifier une adresse IP. Lors de la connexion, le serveur demande nécessairement de s'identifier. Il y a deux possibilités :
— soit le serveur vous connaît via un *login*,
— soit vous êtes inconnu, auquel cas il faut vous connecter en tant qu'anonyme.
Naturellement, un accès anonyme (*login* → `anonymous`) se traduit par des droits réduits.

— Liste des commandes disponibles

```
ftp> help
```

Les manipulations à effectuer sont :

1. vérifier que la partie client est installé, l'installer si ce n'est pas le cas (trouver le nom d'un client via `apt-cache search ftp | more`) ;
2. se connecter anonymement sur le serveur ftp de la machine `172.20.20.16`, puis passer en `Passive Mode` en tapant `pass` ;
3. télécharger le fichier `welcome.msg` (utiliser `help` pour trouver la commande adéquate) ;
4. se déconnecter, puis installer `filezilla` ;
5. lancer une capture des communications avec `wireshark`, puis utiliser `filezilla` pour télécharger le fichier `welcome.msg` qui se trouve également dans le même répertoire ;
6. arrêter la capture et analyser les communications induites par `filezilla` ;
7. refaire la capture des communications en utilisant `filezilla`, mais en se connectant cette fois avec le *login* `ftp-tp` avec le *password* `CqriT` ;

8. renommer le fichier `welcome.msg` que vous avez récupéré précédemment en `welcome-X.msg` où `X` est le numéro de votre machine, puis essayer de le déposer dans le compte `ftp-tp` sur le serveur ftp.

2 Solution sécurisée indispensable avec un *login / password*

Les outils classiques `telnet` et `ftp` présentent un inconvénient majeur. En effet, ils transmettent le *login* et le *password* en clair à la machine distante (en tout cas en les installant de façon basique tel qu'on vient de le faire). Or cela signifie que le couple *login / password* est aisément lisible sur n'importe quelle machine se trouvant sur la liaison entre la machine locale et la machine distante. Ainsi, une personne mal intentionnée utilisant un programme observant le trafic réseau (un renifleur) tel que `wireshark` aura accès aux informations d'authentification. Utiliser les outils classiques, à l'exception de `ftp` en mode anonyme, est le meilleur moyen de se faire pirater.

C'est pour pallier ce problème de sécurité qu'on utilise ***s**ecure **sh**ell* qui est un protocole de niveau Application (modèle client / serveur). Celui-ci permet de travailler à distance sans danger pour le compte distant.

- `ssh` permet de se connecter à distance sur une machine, d'exécuter des commandes, ainsi que de transférer des fichiers entre machines.
- Avec `ssh` un tunnel sécurisé est créé entre les machines. De sorte que ni le couple *login / password*, ni les données ne peuvent être lues ou modifiées pendant le transit à travers un réseau (notamment Internet).
- Par abus de langage, `ssh` désigne à la fois le protocole de chiffrement des communications entre les deux machines et les différents programmes qui l'utilisent.
- Le tableau ci-après décrit les correspondances entre commandes non sécurisées et leurs équivalents `ssh`.

Commande non sécurisée	Équivalent ssh
`telnet`, `rlogin`	`slogin`
`rsh`	`ssh`
`rcp`	`scp`
`ftp`	`sftp`

- `ssh` utilise deux types de cryptographie :
  1. la cryptographie asymétrique à des fins d'authentification ;
  2. la cryptographie symétrique pour chiffrer / crypter les données.

① Cryptographie asymétrique

Ce type de cryptographie associe à un interlocuteur une paire de clés qui sont liées : une clé privée et une clé publique. Si on crypte quelque chose avec l'une, seule l'autre pourra être utilisée pour le décrypter. La clé publique est diffusée librement, elle est utilisée pour crypter les données.

② Cryptographie symétrique

La cryptographie symétrique utilise elle une seule clé (secrète), qui doit donc être connue par les deux interlocuteurs. Cette clé permet de crypter / décrypter les données. L'avantage de la cryptographie symétrique est son moindre coût en ressource processeur. L'inconvénient est l'échange de la clé secrète, c'est là un des rôles de la cryptographie asymétrique. Différents algorithmes de cryptage symétrique sont disponibles dans `ssh`, on peut citer 3des, cf. le manuel.

2.1 Se connecter et exécuter des commandes en mode texte

Pour se connecter, c'est relativement simple, on utilise le nom de la machine distante ou son adresse IP :

```
dupond@alpha:~$ ssh 172.20.20.12
```

ou `ssh beta`. Si vous voulez utiliser un *login* différent, il faut utiliser l'option `-l` : `ssh -l login beta` ou `ssh login@beta`, d'où les deux possibilités :

```
dupond@alpha:~$ ssh -l user 172.20.20.12
dupond@alpha:~$ ssh user@172.20.20.12
```

Sans aucune configuration préalable, il faut explicitement confirmer la première connexion vers une machine donnée en répondant `yes` à la question *Are you sure you want to continue connecting (yes/non/[fingerprint]) ?*. En fait, il est demandé à l'utilisateur s'il veut ajouter la clé publique de la machine "serveur" dans son trousseau (sa base de données de clés publiques). Le trousseau est dans le fichier `known_hosts` du répertoire `.ssh`. Cette clé est utilisée pour authentifier le serveur et sécuriser l'échange de la clé symétrique.

La figure 1 décrit les étapes aboutissant à la mise en place d'un lien sécurisé entre les deux machines lors de la première connexion. Pour les connexions ultérieures, l'étape 1 disparaît. Dès lors que le lien est en place, il ne reste alors plus qu'à s'authentifier sur la machine distante. Deux méthodes d'authentification sont disponibles :

— l'authentification par mot de passe ;
— l'authentification par clé.

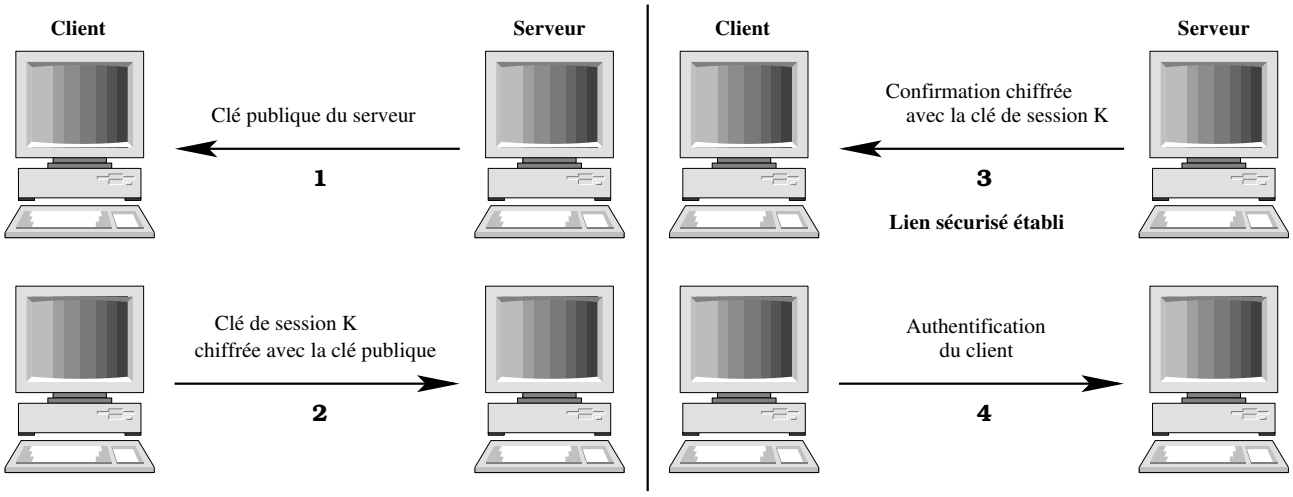

FIGURE 1 – Les différentes étapes de la première connexion `ssh`.

- **Authentification par mot de passe**

Il suffit de taper le *password* associé au *login* sur la machine distante.

- **Authentification par clé**

En lieu et place du mot de passe, l'utilisateur peut s'authentifier auprès du serveur en utilisant la cryptographie asymétrique via une paire de clés privée / publique. Il s'agit d'utiliser le même mécanisme d'authentification que celui utilisé par le serveur. Cette procédure se fait en deux étapes :

1. génération des clés ;
2. distribution de la clé publique aux machines distantes.

① Génération des clés

Il faut utiliser la commande `ssh-keygen`, voici un exemple de génération d'un couple de clés ECDSA :

```
dupond@alpha:~$ ssh-keygen -t ecdsa
```

Par défaut, les clés privée et publique ont une longueur de 256 bits et elles sont stockées respectivement dans les fichiers `id_ecdsa` (permission 600) et `id_ecdsa.pub` (permission 644) du répertoire `.ssh`. Lors de la création, une *passphrase* vous est demandée, il s'agit en quelque sorte d'un mot de passe utilisé pour sécuriser l'accès à la clé privée. Cette *passphrase* vous sera demandée à chaque utilisation de la clé privée, soit à chaque fois que vous vous connecterez. La *passphrase* peut être changée par la commande `ssh-keygen`. Heureusement, il est possible d'éviter d'avoir à répéter la saisie de la *passphrase*, grâce au `ssh-agent` qui tourne en tâche de fond et va mémoriser la clé privée en mémoire. Pour obtenir la clé, le client `ssh` interrogera alors l'agent. Pour donner la clé au `ssh-agent`, il faut utiliser la commande `ssh-add`.
Cela se fait comme suit :

— en mode console (dans l'exemple le shell est de type `bash`)

```
dupond@alpha:~$ ssh-agent bash
dupond@alpha:~$ ssh-add
```

— en mode graphique (si le serveur graphique n'est pas démarré)

```
dupond@alpha:~$ ssh-agent startx
dupond@alpha:~$ ssh-add
```

Si l'environnement graphique est démarré, il suffit d'utiliser directement `ssh-add`.

② Distribution de la clé publique

Il s'agit de donner la clé publique à une ou plusieurs machines distantes. Cela consiste à copier la clé publique dans le fichier `authorized_keys` du répertoire `.ssh` du compte local sur chaque machine où vous voulez pouvoir vous connecter. La commande pour accomplir cette tâche est `ssh-copy-id` :

```
dupond@alpha:~$ ssh-copy-id -i ~/.ssh/id_dsa.pub login@beta
```

Pour **exécuter une commande à distance** la syntaxe est :

```
dupond@alpha:~$ ssh beta [commande]
```

Par exemple, pour savoir qui est connecté sur la machine distance `www.truc.com` on :

```
dupond@alpha:~$ ssh www.truc.com who
```

2.2 Transférer des fichiers en mode texte

Il y a deux façons :

— utiliser `scp` (`ssh copy`) qui s'utilise de la même manière que `cp`, mais qui permet de copier des fichiers de la machine locale vers une machine distance et inversement. À noter qu'il est possible de préciser deux machines distantes avec plus de configuration et dans ce cas la machine locale où on lance la commande est en quelque sorte un intermédiaire ;

— la version `ssh` de `ftp`, à savoir `sftp`.

Voici juste quelques exemples d'utilisation de la commande `scp` :

— Copie du fichier `prog.cpp` vers le `home` local de toto sur la machine `lambda`

```
dupond@alpha:~$ scp prog.cpp toto@lambda:~/
```

— Idem, mais dans le répertoire `programmation`

```
dupond@alpha~$ scp prog.cpp toto@lambda:~/programmation
```

— Recopie de tous les fichiers avec l'extension `.cpp` de la machine *beta* sur la machine *alpha* dans le répertoire `save`

```
dupond@alpha~$ scp beta:~/*.cpp save
```

Les manipulations à effectuer sont :

1. créer un compte `voisin` avec `adduser` sur votre machine afin de permettre à un de vos camarades de se connecter ;
   IMPORTANT : il faut un compte `voisin` pour chacun des voisins qui voudrait se connecter sur votre machine ;
2. utiliser `ssh` pour vous connecter vous-même à la machine d'un de vos camarades, en s'authentifiant par le mot de passe ;
3. ensuite, vous mettrez en place l'authentification par clé. Vous devrez donc :
   (a) générer vos clés de type ECDSA ;
   (b) distribuer votre clé publique sur la machine où vous voudrez vous connecter.
4. utiliser l'agent pour mémoriser la *passphrase* ;
5. utiliser `ssh` pour exécuter à distance la commande `uname -n`
6. utiliser la commande `scp` pour copier un fichier d'une machine à l'autre.

2.3 Se connecter et exécuter des commandes en mode graphique

Jusqu'à présent nous avons uniquement vu comment travailler à distance en mode texte. Le travail en mode graphique sur la machine distante repose sur le concept d'*export de display*, c'est-à-dire que la fenêtre graphique d'une application lancée sur la machine distante sera exportée sur la machine locale / cliente. La machine locale / cliente enverra quant à elle les données du clavier et de la souris vers le machine distante / serveur. Il existe différentes manières de faire de l'export de display.

`ssh` permet de faire de l'export de display, à condition de configurer le serveur pour l'autoriser. La connexion se fait alors par :

```
dupond@alpha~$ ssh -X login@beta
```

Les manipulations à effectuer sont :

1. utiliser `ssh` pour se connecter à la machine d'un de vos camarades, en activant l'export de display ;
2. lancer le navigateur `firefox` sur la machine distante et constater que l'affichage est bien déporté sur votre machine ;
3. fermer `firefox` et se déconnecter.

2.4 Configuration du client `ssh` pour "simplifier" la connexion

Les informations qui suivent sont issues pour partie de ce lien.

Lorsque l'on lance une connexion via le client `ssh`, celui-ci se configure à partir d'options définies dans l'ordre suivant :

1. les options définies dans le fichier `/etc/ssh/ssh_config` ;
2. les options définies dans le fichier `~/.ssh/config` ;
3. les options définies au niveau de la ligne de commande.

Ainsi, cela signifie qu'une option définie dans `~/.ssh/config` peut être remplacée par une définition au niveau de la ligne de commande.

Une connexion typique avec `ssh` va nécessiter de préciser un nom d'utilisateur et un nom de machine, voire un numéro de port :

```
dupond@alpha~$ ssh john@ssh.example.com -p 2022
```

Pour éviter d'avoir à taper chaque fois cette ligne et plutôt lancer la connexion avec `ssh exemple`, il suffit d'ajouter les lignes suivantes dans `~/.ssh/config` :

```
Host exemple
    Hostname ssh.exemple.com
    User john
    Port 2322
```

Il est possible de redéfinir une option d'une section au niveau de l'appel de la commande `ssh`. Ainsi, si on voulait que l'utilisateur `john` soit remplacé par `root` on écrirait :

```
dupond@alpha~$ ssh  -o "User=root" exemple
```

Le fichier `config` peut contenir plusieurs sections `Host` et plusieurs d'entre elles peuvent être utilisées simultanément pour définir des options. Par exemple, sin on a dans le fichier :

```
Host hobbit
    HostName 172.20.20.10
    User bilbo
    Port 7376
    IdentityFile ~/.ssh/hobbit.key

Host gandalf
    HostName 172.20.20.50

Host * !gandalf
    LogLevel INFO

Host *
    User root
    Compression yes
```

Dans ce cas la commande `ssh hobbit` va appliquer tout d'abord les options de `Host hobbit`, puis celles de `Host * !gandalf` (tout sauf `gandalf`) et enfin pour finir celles de `Host *` qui s'applique tout le temps. Toutefois, pour `Host *`, seule l'option `Compression` s'applique car `User` a déjà été défini.

Les manipulations à effectuer sont : *on suppose que vous êtes l'utilisateur* `tpreseau`

1. on commence par réduire l'accès au répertoire `.ssh` en modifiant les droits d'accès avec la commande `chmod 700 ~/.ssh` (les droits sur le répertoire sont peut être déjà corrects) ;
2. on crée un fichier `config` vide et on modifie les droits d'accès pour les réduire au seul propriétaire

   ```
   touch ~/.ssh/config
   chmod 600 ~/.ssh/config
   ```

3. éditer le fichier `config` avec `nano` et ajouter les lignes suivantes en les adaptant par rapport à votre voisin

   ```
   Host voisin
       Hostname 172.20.20.155
       User voisin
       IdentityFile ~/.ssh/ecdsa.key
       IdentitiesOnly yes
   ```

4. se connecter dans le compte `voisin` chez votre voisin via `ssh voisin` ;
5. vérifier que les commandes ci-dessous permettent de se connecter avec le mot de passe

   ```
   dupond@alpha~$ ssh -o PreferredAuthentications=password voisin
   ```

   ```
   dupond@alpha~$ ssh -o PubKeyAuthentication=no voisin
   ```