TP7 - TP SSH

1 Introduction et premiers pas

1.1 Le protocole SSH

Brièvement, le protocole SSH (Secure SHell) permet à des utilisateurs d'accéder à une machine distante à travers une communication chiffrée (appelée tunnel). L'établissement d'une liaison passe par deux étapes. Le client établit d'abord un tunnel sécurisé avec le serveur en utilisant un couple de clés privée/publique : le serveur possède une clé privée (qui ne change que lorsqu'on réinstalle entièrement la machine) et les clients ont une copie de la clé publique. Le client utilise la clé publique pour chiffrer une clé symétrique dite *de session* qui est utilisée pour chiffrer tout le reste des communications. L'authentification de l'utilisateur peut alors être faite sans que le mot de passe soit transmis en clair sur le réseau.

- Expliquez brièvement la nécessité de l'utilisation de SSH
  et comment ce protocole garantit à la fois la confidentialité, l'authentification (dans les deux sens !) et l'intégrité des données échangées.
  N'hésitez pas à vous faire un petit schéma pour expliquer l'établissement d'une connexion.
- Expliquez pourquoi lorsque vous vous connectez pour la première fois à une machine donnée votre client vous demande si la clé publique récupérée est bien celle du serveur.
  Pourquoi la clé privé du serveur doit-elle rester secrète ?
- ssh est basé sur une architecture client/serveur. Sur quel port écoute le serveur ?
- Essayez de vous connecter sur la machine de votre voisin à l'aide de la commande ssh.

2 Assez des mots de passe ?

Lors de la connexion vers une machine distante, ssh demande à l'utilisateur son mot de passe. Imaginez un administrateur système qui doit exécuter cette opération plusieurs fois par jour... Heureusement, de façon analogue à l'authentification des machines, un utilisateur peut utiliser un couple de clés privée/publique pour s'authentifier auprès d'un serveur.

2.1 On oublie les mots de passe

Voici la marche à suivre :

1. Création d'un couple de clés : `ssh-keygen -t rsa`, utilisez pour l'instant le choix par défaut pour les 3 questions posées. La clé privée se trouve dans `~/.ssh/id_rsa` et votre clé publique est dans `~/.ssh/id_rsa.pub`.
2. Transfert de votre clé publique sur le serveur :
   `ssh username@machine "cat >> ~/.ssh/authorized_keys" < ~/.ssh/id_rsa.pub`
   (`authorized_keys` s'appelle selon les versions `authorized_keys2`). Voyez aussi la commande `ssh-copy-id` qui permet de le faire bien plus simplement. Bien sûr, puisque les *homes* sont les mêmes sur les différentes machines du CREMI, il suffirait de copier/coller directement les fichiers, mais entre votre propre ordinateur et `jaguar`, par exemple, il faudrait réellement effectuer un transfert via ssh, clé USB, voire à la main !
3. Vous pouvez vous connecter sur la machine de votre voisin sans mot de passe !

Nous allons maintenant vérifier que ssh se préoccupe de la sécurité de vos clés :
— Donnez à `~` les droits 711.
— Donnez à `~/.ssh` les droits 711.
— Sur votre machine locale, modifiez les droits du fichier `~/.ssh/id_rsa` (qui contient votre clé privée) en 644. Connectez-vous sur le serveur distant qui contient votre clé publique. Que se passe-t-il ? Pourquoi ?
— Rétablissez les droits de `~/.ssh/id_rsa` . Maintenant, sur le serveur distant, modifiez les droits de `~/.ssh/authorized_keys` en 666. Déconnectez-vous puis reconnectez-vous. Expliquez (N'oubliez pas de rétablir les droits d'origine ensuite).
— Ainsi donc, comment l'administrateur du serveur s'assure-t-il que la clé publique installée dans votre *home* est bien la vôtre ?

Je veux fournir à un collègue un accès `ssh` à ma machine. Quelle est la manière la plus sûre de procéder ?

2.2 Mode paranoïaque

Pour protéger votre clé privée et s'assurer que c'est bien la bonne personne qui utilise la clé, il est possible lors de la création des clés de saisir une passphrase servant à chiffrer la clé privée, et qui sera donc demandée à chaque connexion pour pouvoir l'utiliser. Recommencez la procédure de création des clés en utilisant cette fois-ci une passphrase. N'oubliez pas de déposer la nouvelle clé publique dans votre `authorized_keys` . On avait évité d'avoir à taper un mot de passe à l'aide d'une paire de clés, mais maintenant on doit taper une passphrase ! Mais on a quand même amélioré la sécurité. Pourquoi ? (Quelles données passent par le réseau ? Pourquoi est-ce notamment intéressant pour des ordinateurs portables ?)

Pour éviter d'avoir à retaper la passphrase à chaque connexion, on peut utiliser un agent ssh qui va conserver dans un cache votre clé privée déchiffrée durant toute la durée de votre session sur la machine cliente.

1. Tuez d'abord tout agent que gnome ou autre aurait lancé pour vous : `killall ssh-agent`
2. Initialisez l'agent ssh : `eval ‘ssh-agent‘` (attention, ce sont des apostrophes inverses, obtenues avec altgr-7). Constatez avec `printenv | grep SSH` que cela a défini deux nouvelles variables d'environnement.

3. Ajoutez votre clé au cache de l'agent : `ssh-add`
4. Essayez de vous connecter sur une autre machine. Alors ?
5. Ouvrez une autre fenêtre de terminal. Connectez-vous sur une autre machine. Expliquez pourquoi il n'arrive pas à profiter de l'agent. D'habitude, on démarre l'agent avec la session graphique, et tous les processus héritent des deux variables d'environnement.
6. Essayez de vous connecter en cascade (*i.e.* un `ssh` à l'intérieur d'un autre `ssh`) sur différentes machines. Est-ce que cela fonctionne toujours ?

   Remarquez l'option `-A`.

3 Transferts de fichiers

Les commandes `scp` permettent de transférer des fichiers par l'intermédiaire de `ssh`. Par exemple,

```
scp monfichier lamachine:
scp lamachine:lerepertoire/unautrefichier .
```

Testez !

À noter que le répertoire de base pour les chemins distants est `$HOME`

Essayez aussi de transférer un fichier entre deux ordinateurs distants tout en étant sur un troisième. Il faut par contre penser à forwarder l'agent avec `-o ForwardAgent=yes` pour que la source puisse se connecter à la destination.

En pratique, `scp` n'est pas très efficace pour transmettre de nombreux fichiers (option `-r`). Expliquez comment la commande suivante fonctionne et l'importance de l'antislash devant le `;` :

```
tar c myproject | ssh monserveur cd foobar \; tar x
```

4 Affichage X distant

Comme vu au TP2, le protocole `ssh` permet aussi d'exporter un affichage X11 vers votre machine cliente. Quelle option faut-il ajouter à la commande `ssh` ?

Essayez. Connectez-vous sur plusieurs machines en cascade. Est-ce que l'export de l'affichage suit ?

De façon plus générale, le comportement de `ssh` peut être paramétré en créant un fichier `config` dans le répertoire `.ssh` de votre machine client (`man 5 ssh_config`). Refaites les dernières manipulations en utilisant cette fois le fichier de configuration.

5 Redirection de ports

Ssh peut aussi servir à transférer des données pour des services TCP/IP en utilisant une redirection de port. Cela est le plus souvent utile pour accéder à un service qui n'est disponible que depuis certaines machines *internes*.

Par exemple, un serveur tourne sur le port TCP 12345 de `mcgonagall`. Essayez de vous y connecter en lançant depuis votre machine

`nc mcgonagall 12345`

Essayez directement depuis mcgonagall en utilisant

`nc localhost 12345`

Observez à l'aide de `netstat -an` que l'écoute sur le port `12345` n'est effectivement faite que pour l'adresse `localhost` . Au passage, remarquez qu'avec l'option `-n` on obtient les adresses IP sous forme numérique, et que `0.0.0.0` signifie `*` .

L'option `-L` de `ssh` permet de rediriger un port de votre machine vers le port local `12345` de `mcgonagall` (on appelle cela "pont SSH"). L'idée est qu'en donnant l'option `-L port:host:hostport`, toutes les connexions que vous faites vers le port `port` de votre propre machine sont redirigées par `ssh`, pour être effectuées depuis la machine à laquelle `ssh` est connecté, vers la machine `host` et le port `hostport`. On utilise souvent en plus les options `-N` et `-f`. Effectuez un tel pont, et connectez-vous ainsi depuis votre propre machine via le pont SSH.

Refaites la même chose mais en mettant plusieurs machines entre `mcgonagall` et votre machine.

Le serveur NNTP de l'université (`news.u-bordeaux.fr`) n'est accessible sur le port 119 que lorsque l'on est à l'intérieur du réseau de l'université. Comment établir un pont pour pouvoir connecter un client NNTP depuis l'extérieur ?

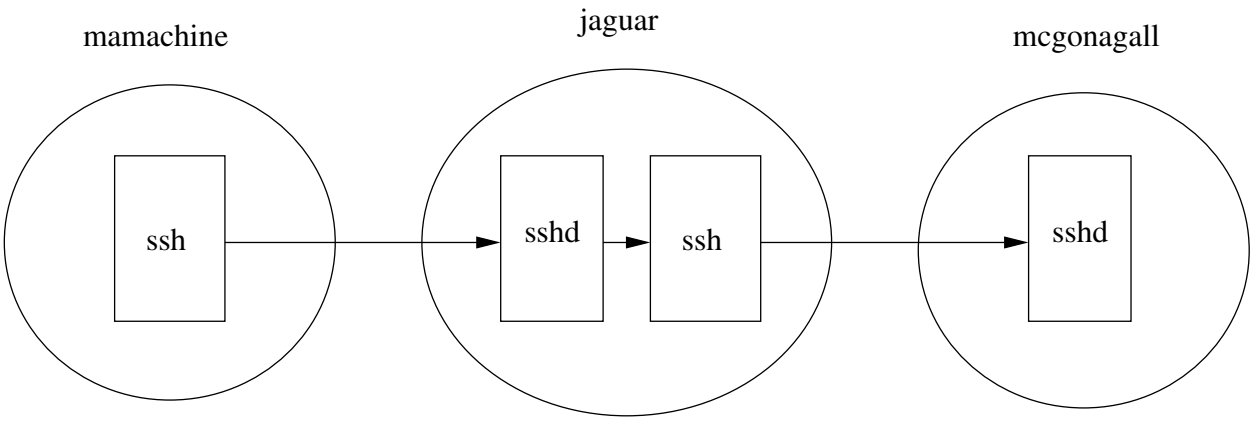
L'option `-R` est le *dual* de `-L`. Que fait-elle ? Faites un schéma.

Notez également que l'on peut ajouter à la volée des redirections de ports : tapez `<Entrée>~C` pour récupérer l'invite `ssh>`, où vous pouvez taper `?` pour la syntaxe.

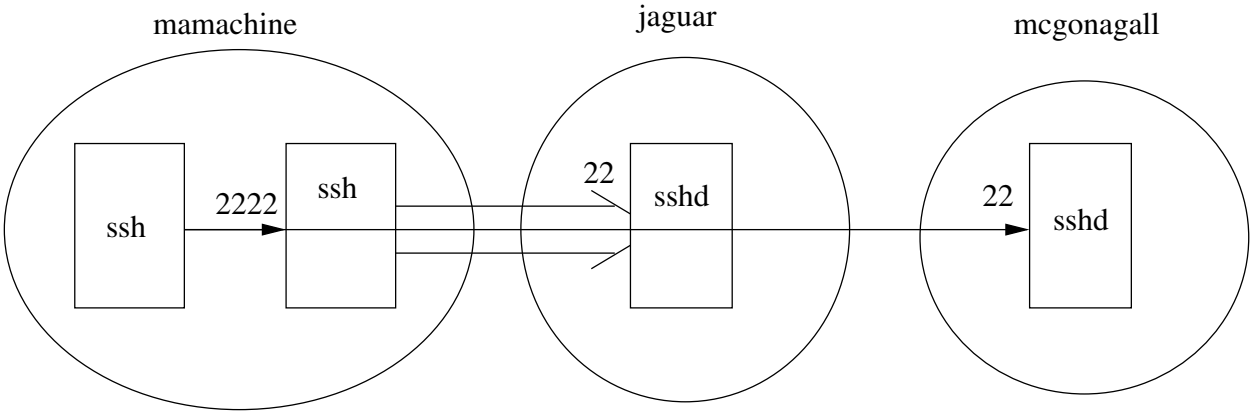
6 Comment cascader du `ssh`

Supposons que vous être à l'extérieur de l'université et vous souhaitez vous connecter à `mcgonagall`, mais comme seule `jaguar` est visible depuis l'extérieur, il est nécessaire de passer par elle. Il y a quatre solutions.

La plus simple est de lancer `ssh jaguar` et à l'intérieur, de lancer `ssh mcgonagall`. C'est un peu pénible, et pour les redirections de ports c'est encore plus pénible.



Une deuxième solution est d'effectuer une redirection de port : lancer un premier ssh vers `jaguar` qui ouvre sur votre machine une redirection (disons sur le port 2222) vers le port ssh de `mcgonagall`. Un simple `ssh localhost -p 2222` depuis votre machine dans un autre terminal suffit alors à se connecter au `sshd` de `mcgonagall` via `jaguar`.



Une troisième solution est d'utiliser l'option `-J` :

```
ssh -J jaguar mcgonagall
```

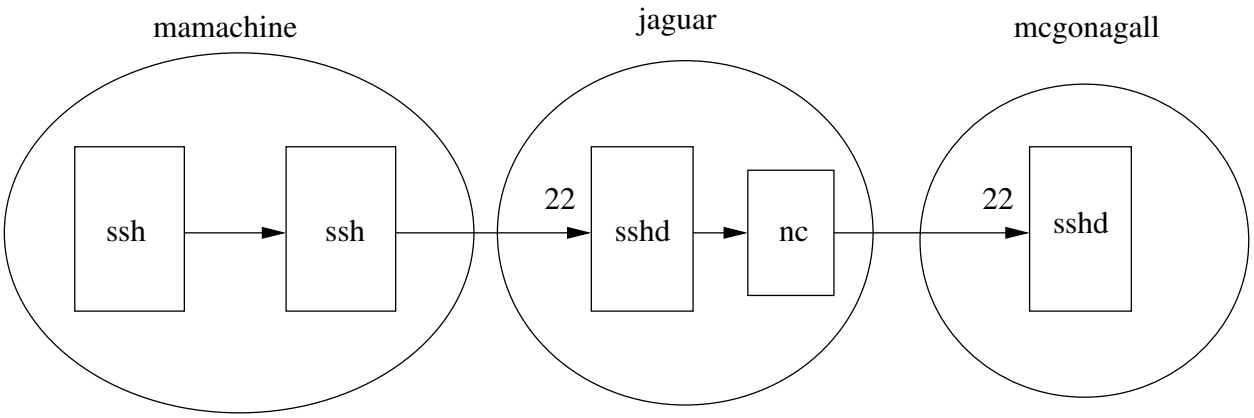
qui effectue automatiquement la deuxième solution :)

Une quatrième solution est d'utiliser `ProxyCommand` : dans votre `~/.ssh/config`, ajoutez les lignes suivantes :

```
Host monmcgonagall
    ProxyCommand ssh jaguar.emi.u-bordeaux.fr -W mcgonagall:22
```

Ces deux lignes indiquent que lorsque l'on lance `ssh monmcgonagall`, plutôt qu'ouvrir une connection TCP, `ssh` va lancer un autre `ssh` vers `jaguar`, le faire se connecter au port `ssh` de `mcgonagall`, et utiliser ce canal pour établir une session `ssh` avec `mcgonagall`

Mettez en œuvre ces quatre solutions.


mamachine
jaguar
mcgonagall
ssh
ssh
22
sshd
nc
22
sshd