Utilisation avancée de ssh

⚠ Une grande partie des exercices proposés (sauf les derniers) ont déjà vus en partie dans les séances précédentes.

1 Présentation du protocole SSH

Le protocole SSH (*Secure Shell*[1]) offre un service de connexion sécurisée à distance (port 22 en TCP/IP). Il est (souvent) basé sur un serveur (`sshd`) et un logiciel client (la commande `ssh`). Ce service remplace les anciens protocoles de connection à distance comme `telnet` , `rsh` ou `rlogin` . Avec ces derniers les informations transitent en clair sur le réseau ce qui permet leur interception.

L'authentification dans SSH peut se faire par mot de passe ou par l'utilisation de clefs asymétriques. A cause de l'utilisation de mot de passe le protocole est sensible aux attaques brutales (tentatives de connexions multiples basées sur des dictionnaires de mots de passe).

Note : Le protocole SSH ne **doit pas** être ouvert sur l'internet sans limitation d'accès. Pour le protéger, vous devez utiliser un logiciel comme DenyHosts [a] (fermeture des accès après trois tentatives) ou mettre en place un service de Port Knocking [b] (ouverture du port SSH à la demande sur la base d'une procédure définie à l'avance, c'est donc un secret partagé entre l'administrateur du serveur et son utilisateur).

a. http://denyhosts.sourceforge.net/
b. http://fr.wikipedia.org/wiki/Port_knocking

2 Vérification du service SSH

1. Vérifiez la présence des packages :

```
yum list "*openssh*"
```

2. Démarrez le service `sshd` :

```
systemctl restart sshd
```

3. Vérifiez le :

```
systemctl status sshd
netstat -tap | grep ssh
```

4. Testez le :

```
ssh -v localhost
```

1. http://fr.wikipedia.org/wiki/Secure_Shell

3 Redirection du trafic X

Note : Un utilisateur connecté de manière distante a souvent besoin d'utiliser des logiciels graphiques. Pour ce faire, il va être amené à mettre en place une liaison client/serveur entre son serveur X (son écran graphique) et son logiciel (le client X). Malheureusement cette liaison n'est pas protégée et son travail peut donc être observé. Pour éviter ce problème, le protocole SSH permet d'encapsuler le trafic X dans le tunnel sécurisé.

- Commencez par lire les fichiers de configuration du serveur `sshd` (fichier `/etc/ssh/sshd_config`) et du client `ssh` (fichier `/etc/ssh/ssh_config`). Vérifiez que la redirection de X est acceptée par le serveur et le client.
- Installez sur la machine invitée le package `xorg-x11-xauth`.
- Testez la redirection X en vous connectant (à partir de votre machine hôte DOSI) sur votre machine invitée (avec la configuration ssh mise en place lors du premier TP) :

  ```
  ssh -X VM
  ```

  Sur la machine invitée, observez la valeur de la variable d'environnement `DISPLAY`. Elle indique le faux serveur X géré par `ssh`.
- Lancez une commande graphique (`nedit`).

4 Port forwarding avec ssh

Note : Soient trois machines (`M1`, `M2`, et `M3`) la votre étant la première. Nous voulons atteindre, depuis `M1` un service de `M3` qui n'est ouvert qu'à `M2`.

- Lancez `thttpd` sur votre VM.
- Nous voulons nous connecter sur le serveur WEB de votre VM à partir de la machine hôte DOSI.
- À partir de la machine hôte, connectez-vous sur votre VM en demandant l'ouverture d'un tunnel IP du port `9000` de la machine hôte vers le port web (`80`) de la VM (`10.0.2.15`).

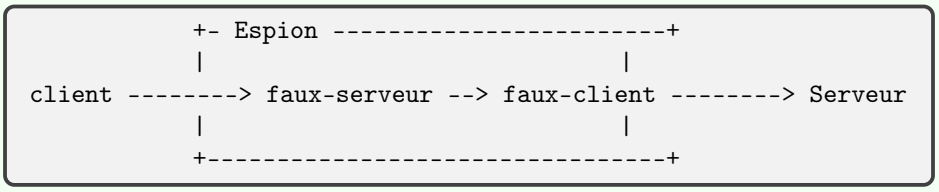
  ```
  ssh VM -L 9000:10.0.2.15:80
  ```
- Avec `netstat` [2] vérifiez que le processus client `ssh` écoute bien le port 9000 sur la machine hôte. Tentez ensuite de vous connecter avec un navigateur exécuté la machine hôte sur l'adresse :

  ```
  http://localhost:9000
  ```

2. https://man7.org/linux/man-pages/man8/netstat.8.html

5 Authentification par clé publique/privée

Note : Malgré le mécanisme de cryptage, le protocole SSH est sensible à une attaque du type *Man In the Middle*. Dans cette situation, le client dialogue (sans le savoir) avec un faux serveur qui, lui-même, renvoie les données vers le serveur (qui pense les recevoir du vrai client). Lors de ces échanges le mot de passe envoyé par le client va être capté et enregistré par l'espion.



Pour éviter que le mot de passe ne transite par le réseau, nous allons mettre en place une authentification basée sur un système de clef publique (installée sur le poste serveur) et de clef privée (installée sur le poste client).

Vous pouvez lire cette page [3] ou chercher sur google d'autres ressources pour comprendre la cryptographie à clé publique.

5.1 Authentification par clés

- Pour cet exercice, vous devez travailler avec plusieurs comptes (une compte **source** et un **destination**). **Conseil** : vous pouvez utiliser votre compte sur la machine DOSI et le compte `root` de votre VM.
- Sur la machine hôte (la source), commencez par générer une pair de clés avec la commande `ssh-keygen` [4].
- Repérez la clé publique (fichier `$HOME/.ssh/*.pub`) et copiez-la (avec `ssh-copy-id`) dans le fichier `$HOME/.ssh/authorized_keys` de la machine de destination (votre VM) :

```
ssh-copy-id -i .ssh/votre-cle.pub root@VM
```

- Vérifiez ensuite que vous pouvez vous connecter sur le compte destination en donnant le **mot de passe qui chiffre la clé privée de votre compte source** et non pas le mot de passe du compte destination.
- **Très important** : lors de la construction des clefs, vous devez absolument donner une *passphrase* pour chiffrer votre clef privée. Si cette clef n'est pas chiffrée, alors un pirate s'introduisant sur la machine cliente peut récupérer la clef et donc, **se connecter sur les postes serveurs sans donner de mot de passe**.

5.2 Utilisez l'agent d'authentification

Vérifiez que le démon `ssh-agent` (ou un autre processus du même type) est bien accessible par tous les processus de votre session en observant les variables d'environnement :

```
printenv | grep -i SSH
```

Ajoutez votre identité au cache géré par `ssh-agent` avec la commande `ssh-add`. Vérifiez ensuite cette ajout avec « `ssh-add -l` ». Vous devez maintenant être capable de vous connecter sur le compte de destination **sans avoir à donner le mot de passe de votre clé privée**.

3. https://www.bibmath.net/crypto/index.php ?action=affiche&quoi=moderne/clepub
4. https://man7.org/linux/man-pages/man1/ssh-keygen.1.html

6 Copie de fichiers/répertoires par scp

`ssh` inclut via `scp` [5] la possibilité de copier des fichiers et/ou des répertoires d'une machine sur une autre et d'un compte à une autre).

⏩ **Travail à faire :** Lisez la page de manuel de `scp` [a] et utilisez cette commande pour copier un fichier de la machine hôte vers la machine invitée.

a. https://man7.org/linux/man-pages/man1/scp.1.html

7 Synchronisation de répertoires par ssh

Utilisez la commande `rsync` [6] pour copier un répertoire d'une machine à une autre :

```
dnf -y install rsync

rsync -av -e ssh répertoire-source \
    utilisateur@machine-destination:répertoire-destination
```

ℹ **Note :** Cette commande, particulièrement puissante, réalise les copies en mode incrémental en mettant à jour la destination par rapport à la source.

8 Monter des répertoires à distance via ssh

Utilisez la commande `sshfs` [7] pour monter sur votre machine un répertoire distant :

```
# le logiciel nécessaire
dnf -y install sshfs

# création du point de montage
mkdir /mnt/test

# montage
sshfs etud@localhost:. /mnt/test
mount | grep etud

# exploitation
ls -l /mnt/test

# démontage
umount /mnt/test
```

ℹ **Note :** Il faut que noter que `sshfs` fonction également avec des clefs publiques/privées ce qui permet d'éviter la phase des mots de passe.

5. https://man7.org/linux/man-pages/man1/scp.1.html
6. https://man7.org/linux/man-pages/man1/rsync.1.html
7. https://man7.org/linux/man-pages/man1/sshfs.1.html