SSH et compagnie : sftp, scp et ssh-agent

Linux Gazette numéro 64

Matteo Dell'Omodarme

matt@martine2.difi.unipi.it

Le but de cet article est de faire une introduction à quelques programmes utiles dans la suite **SSH**, à savoir **sftp**, **scp**, **ssh-agent**, et **ssh-add**. Nous supposerons dans ce qui va suivre que le daemon **sshd2** est bien configuré et lancé.

1. sftp et vue d'ensemble de scp

Concentrons-nous sur **sftp** et **scp**.

Le premier (**sftp** - en anglais, protocole de transfert de fichier sécurisé) est un client à la mode **ftp** utilisé pour transférer des fichiers à travers le réseau. Il n'utilise pas de daemon FTP (**ftpd** ou **wu-ftpd**) pour les connexions, ce qui permet une amélioration significative de la sécurité du système. En fait, en surveillant les fichiers journaux de nos systèmes, nous avons noté que 80% des attaques du mois dernier étaient lancées

contre le daemon **ftpd**. L'utilisation de **sftp** empêche tous ces essais puisqu'elle permet d'arrêter le daemon **wu-ftpd** potentiellement dangereux.

La seconde (**scp** - en anglais, copie sécurisée) est employée pour copier des fichiers à travers le réseau de façon sécurisée. C'est un remplacement pour la peu sûre commande **rcp**.

sftp et **scp** n'exigent aucun daemon dédié puisque les deux programmes sont liés au serveur **sshd**. Afin d'utiliser **sftp** et **scp** vous devez insérer la ligne suivante dans le fichier de configuration `/etc/ssh2/sshd2_config` :

```
subsystem-sftp                  sftp-server
```

Vous devez redémarrer sshd après cette modification. Vous pourrez ainsi utiliser **sftp** et **scp** sur les systèmes où **sshd** est en fonction.

1.1. sftp

sftp utilise SSH2 pour ses connexions, le transport du fichier bénéficiant ainsi de la meilleure sécurité possible. Il y a deux avantages à utiliser **sftp** au lieu de **ftp** :

- Les mots de passe ne sont jamais transférés en clair, empêchant n'importe quelle attaque de type écoute passive.
- Les données sont chiffrées pendant le transfert, rendant difficile l'espionnage ou la modification de la connexion.

L'utilisation de **sftp2** est simple. Supposons que vous vous connectiez à votre compte mon_nom sur la machine serveur1. Pour ce faire, utilisez la commande :

```
sftp mom_nom@serveur1
```

Quelques options peuvent être spécifiées sur la ligne de commande (voir la page de manuel afférante à **sftp** pour les détails complets).

Quand **sftp2** est prêt à accepter des commandes utilisateur, il envoie une invite *sftp*>. Vous trouverez dans la page de manuel de **sftp** une liste complète des commandes que l'utilisateur peut utiliser, avec parmi elles :

quit	Quitte l'application
cd {*répertoire*}	Change le répertoire de travail courant distant
lcd {*répertoire*}	Change le répertoire de travail courant local
ls [-R] [-l] *fichier*...	Renvoie la liste des fichiers disponibles sur le serveur. Si l'argument est un répertoire, son contenu est renvoyé. Quand l'option `-R` est spécifiée, l'arborescence est renvoyée récursivement (par défaut, les sous-répertoires des répertoires en argument ne sont pas visités.) Quand l'option `-l` est indiquée, les permissions, propriétaires, tailles et heures de modifications de fichier sont aussi montrées. Quand aucun argument n'est spécifié, le contenu du répertoire courant est renvoyé. Les options `-R` et `-l` sont mutuellement incompatibles.
lls [-R] [-l] *fichier*...	idem ls, mais sur les fichiers et répertoires locaux.
get *fichier*...	Transfère les fichiers indiqués spécifiés du serveur vers le client. Les répertoires sont récursivement copiés avec leur contenu.
put *fichier*...	Transfère les fichiers indiqués spécifiés du client vers le serveur. Les répertoires sont récursivement copiés avec leur contenu.
mkdir *répertoire*...	Essaye de créer le répertoire dont le nom est indiqué en argument

rmdir *répertoire*...	Essaye de détruire le répertoire dont le nom est indiqué en argument

sftp2 supporte l'utilisation de caractères jokers en paramètre des commandes **ls**, **lls**, **get** et **put**. Le format en est défini dans la page de manuel sshregex.

Puisque **sftp** utilise des techniques de chiffrement, il existe un inconvénient : la connexion est plus lente (d'un facteur de 2 ou 3 au vu de mon expérience), mais ce point est d'un intérêt marginal considérant les grands avantages de sécurité apportés.

Dans un essai conduit sur notre réseau local, un renifleur de réseau pouvait récupérer une moyenne de 4 mots de passe à l'heure sur les connexions FTP. La mise en oeuvre de **sftp** en tant que protocole standard pour le transfert de fichier à travers le réseau a pu éliminer ce problème de sécurité.

1.2. scp

scp2 (copie sécurisée) est utilisé pour copier des fichiers à travers le réseau de manière sécurisée. Il utilise SSH2 pour le transfert des données : il utilise la même authentification et fournit la même sécurité que SSH2.

C'est probablement la manière la plus simple de copier un fichier sur une machine à distance. Supposons que vous voulez copier le fichier `nom_fichier` contenu dans le répertoire `rep_local` au répertoire `rep_distant` de votre compte mon_nom du serveur serveur1. En utilisant **scp**, vous pouvez taper la ligne de commande :

```
scp rep_local/nom_fichier mon_nom@serveur1:rep_distant
```

Le fichier est ainsi copié en gardant son nom. Des caractères jokers peuvent être utilisés (à ce sujet, consulter la page de manuel de sshregex).

La commande :

```
scp rep_local/* mon_nom@serveur1:rep_distant
```

copie tous les fichiers du répertoire `rep_local` dans le répertoire `rep_distant` de la machine serveur1.

La commande:

```
scp mon_nom@serveur1:rep_distant/nom_fichier .
```

copie le fichier `nom_fichier` du répertoire distant sur serveur1 `rep_distant` sur le répertoire local courant.

scp supporte bon nombre d'options et permet des copies entre deux systèmes distants comme dans l'exemple suivant :

```
scp mon_nom@serveur1:rep_distant/nom_fichier mon_nom@serveur2:autre_rep
```

Voyez la page de manuel pour une présentation complète.

Évidemment, en utilisant **scp**, vous devez connaître l'arborescence exacte des répertoires de la machine distante, si bien que dans la pratique **sftp** est souvent préféré[1].

2. Gestion des clés de SSH

La suite SSH contient deux programmes permettant de gérer les clés d'authentification, permettant à l'utilisateur de se connecter sans saisir de mot de passe ou de phrase de passe. Ces programmes sont **ssh-agent** et **ssh-add**.

2.1. ssh-agent

Dans la page de manuel de **ssh-agent**, nous pouvons lire : « ssh-agent2 est un programme pour conserver des clés privées d'authentification. L'idée est que ssh-agent2 est démarré au début d'une session X-Window ou d'une session de connexion terminal, et que tous les autres programmes ou fenêtres sont lancés en tant

que processus fils de ssh-agent2 (la commande lance X ou l'interpréteur de commande interactif de l'utilisateur). Les programmes lancés par l'agent héritent de la connexion à l'agent, et l'agent est automatiquement utilisé pour l'authentification à clé publique lors de connexions à d'autres machines utilisant ssh. »

Il y a deux manières d'utiliser **ssh-agent** selon que vous utilisiez **xdm** ou non. Dans le premier cas, il vous faudra modifier votre fichier *.xsession* de votre répertoire utilisateur (`$HOME`). Deux procédures pour y arriver : copiez `.xsession` en `.xsession.toto` et mettez uniquement la ligne qui suit dans `.xsession` :

```
exec ssh-agent ./.xsession-toto
```

Vous pouvez aussi modifier `.xsession` et chercher chaque ligne contenant l'expression « exec programme ». Modifiez ces lignes pour qu'elles apparaissent sous la forme « exec ssh-agent programme »

Quittez votre session X et redémarrez-la. **ssh-agent** démarrera la session X en tant que processus fils unique et attendra les clés SSH à ajouter à sa base de données.

Si **xdm** ne tourne pas sur votre machine, la procédure est plus simple car vous n'aurez qu'à lancer votre session X en utilisant la commande :

```
ssh-agent startx
```

Ainsi, **ssh-agent** s'exécutera correctement.

2.2. ssh-add

Dès que **ssh-agent** est en place, vous pouvez ajouter des identités à sa base de données en utilisant la commande **ssh-add**. Vous ne pouvez ajouter ces identités que depuis des processus fils d'un ancêtre **ssh-agent**, sinon le message d'erreur suivant vous sera affiché : `Failed to connect to authentication agent - agent not`

`running ?` L'utilisation de **ssh-add** est simple : lancez la commande suivante :

```
ssh-add
```

ssh-add balaiera le fichier `$HOME/.ssh2/identification` qui contient les noms des clés privées utilisées pour l'authentification. Si ce fichier n'existe pas, le nom standard de la clé privée est utilisé (à savoir : `$HOME/.ssh2/id_dsa_1024_a`.) Si une clé publique requiert une phrase de passe, **ssh-add** la demande à l'utilisateur : `Adding identity: /home/matt/.ssh2/id_dsa_1024_a.pub Need passphrase for /home/matt/.ssh2/id_dsa_1024_a (..) Enter passphrase:` Vous pouvez obtenir la liste de toutes les identités endossées par l'agent en utilisant la commande **ssh-add `-l`** : `Listing identities. The authorization agent has one key: id_dsa_1024_a: 1024-bit dsa, (...)`

3. Conclusion, liens utiles

Beaucoup d'utilisateurs de **telnet**, de **rlogin**, de **ftp** ne se rendent pas compte que leur mot de passe est transmis en clair à travers le réseau, mais il l'est. L'utilisation de protocoles sécurisés a pû permettre une transmission sécurisée à travers des réseaux non-sûrs.

SSH, en chiffrant le trafic, élimine pertinemment l'écoute clandestine, le détournement de connexion, et d'autres attaques réseau.

Ces articles sont seulement une introduction à SSH ; vous trouverez plus d'informations à ce sujet dans les pages de manuel de **ssh**, de **sshd** et de **sftp**.

Vous pouvez obtenir la suite SSH de www.ssh.com/products/ssh/ (http://www.ssh.com/products/ssh), site principal de SSH ou d'un site miroir. Vous y trouverez également quelques informations très intéressantes sur la technologie de SSH et la cryptographie en général dans le Tech corner (http://www.ssh.com/tech/).

Vous pouvez aussi visiter www.openssh.com (http://www.openssh.com/) où vous pouvez télécharger OpenSSH, mise en oeuvre libre du protocole SSH. La version

portable est disponible à www.openssh.com/portable.html (http://www.openssh.com/portable.html). Vous pouvez également lire la FAQ d'OpenSSH : www.openssh.com/faq.html (http://www.openssh.com/faq.html).

Paru dans le numéro 64 de la Linux Gazette, mars 2001.

Traduction française par Jérôme Fenal <jerome@fenal.org>.

Notes

1. NdT : disons plutôt que **scp** sera utilisé pour lancer des transferts programmés avec **cron**, par exemple, et que **sftp** sera utilisé en mode interactif.