Créer un serveur sftp chrooté

Author : pierrehirel

Les serveurs et clients ssh et sftp sont facilement installables sur les distributions GNU/Linux modernes. Cependant par défaut les deux ne sont pas dissociables : les utilisateurs ayant un compte sur le serveur peuvent accéder à la fois au service sftp et à ssh. Or, un accès ssh permet potentiellement d'avoir accès à tout le disque dur, ainsi qu'à tous les programmes installés. Nous verrons ici comment cantonner les utilisateurs à l'utilisation du protocole sftp pour le transfert de fichiers uniquement (l'accès en ligne de commandes via ssh sera interdit), et à certains dossiers seulement.

1. Mettre en place un serveur ssh et sftp

Pour bénéficier des fonctionnalités sftp, il suffit d'installer *ssh-server* sur le serveur. En effet, le protocole sftp est inclus dans ce serveur.

Pour s'y connecter il faut alors utiliser la commande *sftp* :

client:$ sftp -oPort=2222 [user]@[server-address]

-o permet de passer des options *ssh* à la commande *sftp* (ici c'est le port ssh qui est changé), et bien sûr **[user]** est à remplacer par votre nom d'utilisateur et **[server-address]** par l'adresse IP du serveur (ou son URL). Il est aussi possible d'utiliser un client FTP gérant le SFTP, comme Filezilla par exemple.

2. rssh : créer un environnement restreint (chroot)

Le serveur sftp mis en place précédemment n'est pas dissocié de l'accès ssh. Ainsi en l'état, si on souhaite créer un compte pour d'autres utilisateurs (amis, collègues…), alors ceux-ci auront, avec leurs identifiants, un accès au sFTP, mais aussi un accès en ssh. Nous verrons dans cette partie comment restreindre l'accès à certains utilisateurs, en leur autorisant l'accès au service sftp, mais en leur interdisant l'accès au service ssh.

La méthode consiste à isoler ces utilisateurs dans un environnement restreint, dit *chroot*. Du fait que les utilisateurs seront enfermés dans cet environnement sans avoir accès au reste du système, on parle parfois de « prison » chroot. Pour la mettre en place, il faut installer le programme *rssh*, puis éditer le fichier */etc/rssh*. Les lignes importantes à modifier sont ici :

/etc/rssh

```
# allowssh
# allowscp
# allowcvs
# allowrdist
# allowrsync
allowsftp

chrootpath = "/chroot"
```

Ici nous interdisons l'accès à tous les services (les lignes commençant par un # sont des commentaires), à l'exception de sFTP (la ligne *allowsftp* est décommentée). La ligne *chrootpath* indique le répertoire de la « prison » chroot. Il est possible dans ce fichier de configurer des droits pour chaque utilisateur (certains utilisateurs auraient droit seulement au sftp, d'autres auraient droit aussi au ssh, etc.) mais nous ne nous y attarderons pas ici.

Ensuite, il faut créer le répertoire du chroot en lui-même, correspondant à ce qui a été déclaré précédemment dans le *chrootpath* :

```
server~:# mkdir /chroot
```

Ensuite il faut peupler ce répertoire avec une copie des programmes et librairies qui seront nécessaires à l'utilisateur. Certains tutos indiquent de copier des fichiers à la main depuis les répertoires système… ce qui est plutôt hasardeux, et il suffit d'oublier un fichier pour faire capoter l'ensemble. Je vous conseille plutôt d'utiliser le script déjà tout fait qui accompagne le programme rssh (sous Debian ce fichier est disponible dans le répertoire */usr/share/doc/rssh/examples/mkchroot.sh* après avoir installé rssh ; pour les systèmes qui n'en disposent pas, ce script est disponible par exemple ici). Il est à utiliser comme suit :

```
server~:# sh mkchroot.sh /chroot
```

où **/chroot** doit bien sûr correspondre au répertoire du chroot précédemment créé. À ce stade, ce répertoire */chroot* est un « système réduit », contenant une arborescence similaire à celle du système (avec des répertoires *etc*, *lib*, *usr*…), et seulement un petit nombre de programmes et de librairies. Si vous (ou les utilisateurs) avez des besoins spécifiques vous pouvez ajouter d'autres programmes dans les répertoires adéquats du chroot.

Pour pousser la sécurité, il est important que les utilisateurs ne puissent pas accéder à ces répertoires créés dans le chroot. Nous le verrons plus bas, les utilisateurs doivent être « enfermés » dans un répertoire, et ne doivent pas pouvoir en sortir. Par exemple si les utilisateurs peuvent voir le contenu de */chroot/etc/passwd*, alors ils peuvent connaître les identifiants de tous les utilisateurs, ce qui n'est pas forcément une bonne chose. Aussi, tous les répertoires du chroot doivent avoir pour propriétaire *root*, et pour groupe *root*. Pour que les utilisateurs ne puissent pas consulter ces dossiers, il faut retirer les droits de lecture et d'écriture, placez-vous dans le dossier */chroot* et tapez :

```
server:~# chown root:root . *
server:~# chmod o-rw . *
```

Ceci empêchera les utilisateurs d'accéder au répertoire */chroot* en lui-même, ainsi qu'aux dossiers de l'environnement chroot.

3. Gestion des partages

La « prison » (le chroot) étant créée, il faut maintenant y ajouter les répertoires que l'on souhaite partager. À titre d'exemple, admettons que nous voulions que les utilisateurs aient un accès (mais un accès restreint, à travers le chroot seulement) au dossier */media/data/Documents* du serveur.

Commençons par créer un dossier « partage » qui sera le point d'entrée des utilisateurs, et contiendra tous les dossiers auxquels les utilisateurs auront accès :

server:~# mkdir /chroot/partage

Ensuite à l'intérieur de *partage*, il faut créer de nouveaux dossiers ; par exemple créons un dossier *Docs*. Ensuite il faut monter le répertoire que nous voulons partager (*/media/data/Documents*) dans le dossier *Docs* que l'on vient de créer dans le chroot :

server:~# mount --bind /media/data/Documents/ /chroot/partage/Docs/

Cette commande permet de reproduire l'arborescence d'un répertoire dans un autre répertoire. Nous aurions aussi bien pu créer un lien symbolique (*ln -s /media/data/Documents*), mais l'utilisation d'un *mount* avec l'option *bind* présente certains avantages. Contrairement à un lien, le répertoire, une fois monté, ne peut pas être supprimé par les utilisateurs ; en vérité il ne peut même pas être supprimé par l'utilisateur root (pour supprimer un tel répertoire, root doit d'abord le démonter, puis le supprimer). Cela évite donc les erreurs de manip, de la part des utilisateurs comme de la part de l'administrateur. Pour chaque nouveau dossier que vous souhaitez partager avec ces utilisateurs, il faut reproduire cette dernière commande.

Et bien sûr, afin de ne pas perdre les points de montage à chaque démarrage, il est conseillé de les ajouter dans le fichier */etc/fstab* :

/etc/fstab

/media/data/Documents /chroot/partage/Docs bind defaults,bind 0 0

4. Gestion des droits sur les fichiers

Le plus pratique est de créer un nouveau groupe dédié à l'accès au sftp. Nommons par exemple ce groupe « sftpaccess » :

server:~# addgroup sftpaccess

Il faut ensuite changer le groupe de tous les fichiers et dossiers du répertoire « Documents » que nous voulons partager (l'option **-R** pour « récursif », indique de propager ces changements à tous les sous-dossiers) :

server:~# chgrp -R sftpaccess /media/data/Documents

Et, si l'on veut que les utilisateurs n'aient un accès qu'en lecture seule (pas de modification ni de suppression de fichiers possibles), il faut supprimer les droits d'écriture pour le groupe :

server:~# chmod -R g-w /media/data/Documents

Il est impossible de détailler ici tous les exemples de gestion des droits utilisateurs. Certains dossiers par exemple peuvent être configurés en « upload », autrement dit les utilisateurs auront le droit d'écrire dedans ou d'y supprimer des fichiers ; cela peut être pratique pour que les utilisateurs échangent facilement des fichiers entre eux par exemple. Charge à l'administrateur de bien configurer les droits dans tous les répertoires partagés.

5. Gestion des utilisateurs

Maintenant que tout est prêt et configuré, il ne reste plus qu'à ajouter des utilisateurs. On commence par ajouter des utilisateurs au système, comme on le ferait pour des utilisateurs « normaux », par exemple pour ajouter l'utilisateur « dupont » :

server:~# adduser dupont

À ce stade l'utilisateur « dupont » est un utilisateur « normal » (non chrooté), et il a un accès ssh complet au serveur. Pour le cantonner à l'environnement chroot, cela se fait en deux temps : premièrement, modifier le chemin d'accès de cet utilisateur (par défaut */home/dupont*) avec le répertoire du chroot (ici */chroot/partage*), et que son shell n'est pas */bin/bash*, mais le shell rssh ; ceci se fait en modifiant la ligne correspondant à cet utilisateur dans */etc/passwd* (les numéros d'utilisateur et de groupe, ici **1007** et **1008**, doivent correspondre à l'utilisateur en question) :

/etc/passwd

dupont:x:1007:1008:,,,:/chroot/partage:/usr/bin/rssh

Et deuxièmement, il faut ajouter cet utilisateur dans l'environnement chroot, en ajoutant la ligne correspondante dans */chroot/etc/passwd* :

/chroot/etc/passwd

dupont:x:1007:1008:,,,::

Pas besoin ici de re-préciser le chemin du répertoire d'accès de cet utilisateur, ni son shell : ces champs sont ignorés puisqu'ils ont déjà été lus auparavant dans */etc/passwd*.

Enfin, il est nécessaire d'ajouter l'utilisateur au groupe qui va bien, pour qu'il possède les bons droits sur les fichiers :

server:~# gpasswd -a dupont sfptaccess

Les instructions de cette section **5.** sont à réitérer pour chaque utilisateur que vous voulez ajouter au chroot.

6. Conclusion

Grâce à rssh, les utilisateurs chrootés n'ont accès qu'à un nombre restreint de services que vous avez choisis. Ce système est pratique pour éviter que les utilisateurs n'aient accès à toute l'arborescence de votre machine ; ils n'ont accès qu'à un petit nombre de répertoire que vous avez choisis et que vous avez bien voulu ajouter dans le chroot.

Appliqué ici à la création d'un serveur sftp à droits restreints, un environnement restreint *chroot* peut avoir bien d'autres applications. Par exemple, avec les explications données ici vous pouvez simplement autoriser le ssh ; les utilisateurs n'auront alors accès qu'aux programmes que vous aurez ajoutés dans le chroot (dans */chroot/usr/bin* par exemple). Autre exemple : dans un environnement 64 bits, on peut imaginer un chroot contenant certaines librairies 32 bits pour faire tourner des applications spécifiques. Ou encore, si un logiciel particulier a besoin de vieilles librairies qui ne sont plus fournies (ou fournies dans des versions non-compatibles) par la distribution, on peut créer un chroot contenant ces vieilles librairies, et restreindre l'exécution du logiciel à cet environnement chroot ; dans ce cas ce ne sont pas les utilisateurs qui sont dans la « prison », mais un logiciel.

Attention toutefois : un chroot n'est pas une solution-miracle invulnérable ! La première faille est l'administrateur lui-même, autrement dit vous : si vous paramétrez mal les droits sur les fichiers (si par mégarde les utilisateurs ont accès au fichier *passwd* par exemple), cela peut compromettre l'intégrité du chroot dans son ensemble. De même, attention aux programmes que vous ajoutez dans l'environnement chroot, notamment les compilateurs et les interpréteurs de scripts : ils peuvent rendre le système vulnérable et permettre à des utilisateurs de s'échapper du chroot pour explorer le reste du système (voir par exemple ceci). La plus grande vigilance est donc recommandée dans l'administration d'un chroot.