Outil keychain

Connexion SSH persistante

par Philippe Beaucart

Date de publication : 22/05/2011

Dernière mise à jour : 14/06/2011

L'objectif de cet article est de mettre en œuvre un outil (*keychain*) permettant de s'affranchir de la saisie manuelle de mots de passe lors de la connexion sécurisée distante depuis un client Linux vers un serveur Linux (*ssh*), tout en conservant la sécurité apportée par l'échange de clés chiffrées. Le seul cas de réinitialisation avec *keychain* est le redémarrage de la machine cliente (un cas plutôt rare sur une machine physique Linux).

Le prérequis est donc de disposer de deux machines Linux, physiques ou virtuelles, dont l'une est configurée avec un serveur SSH et l'autre disposant du client SSH.

I - Introduction....3
II - Prérequis....4
II-A - Installer les paquets nécessaires....4
II-B - Configurer votre client et serveur SSH....4
II-B-1 - Changer le port d'écoute du serveur SSH....4
II-B-2 - Utiliser le protocole SSH version 2....5
II-B-3 - Interdire la connexion SSH en tant qu'utilisateur root....5
II-B-4 - Autres recommandations de sécurité....6
III - Installer et configurer le chainage des clés....8
III-A - Générer un jeu de clés publique et privée....8
III-B - Autoriser la clé publique de A sur le serveur B....9
III-C - Vérifier et tester la connexion SSH....10
III-D - Automatiser le processus de connexion par session....12
III-E - Une connectivité plus « permanente »....12
III-E-1 - Pour la ligne de commande....12
III-E-2 - Pour vos scripts bash....13
III-F - Connexions chainées avec les machines A, B, C....13
IV - Exemple d'utilisation : sauvegarde distante....14
IV-A - Configuration....14
IV-B - Script de sauvegarde (rsync)....14
IV-B-1 - Créez le script de sauvegarde....14
IV-B-2 - Automatiser le lancement du script....16
V - Résumé....17
VI - Documentation....18
VI-A - Algorithmes et clés de chiffrement....18
VI-B - Installer un serveur SSH....18
VI-C - Aspects sécuritaires....18

Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

I - Introduction

Ce guide ou « cookbook » est orienté pour des distributions de type *Debian* (dont *Ubuntu* fait partie), sachant que le principe reste identique sur d'autres distributions Linux. La connexion SSH de chacune des deux machines doit être préalablement configurée et fonctionnelle.

La connexion d'un client SSH vers un serveur SSH est fonction (d'au moins) trois paramètres principaux :

- l'utilisateur distant ;
- l'adresse de la machine distante ;
- le port utilisé pour la connexion.

On appellera « machine A » ou « client » la machine cliente depuis laquelle on cherchera à se connecter sur le serveur SSH de la « machine B ». Les utilisateurs de chaque machine seront nommés respectivement *user_a* et *user_b*.

*Un tutoriel spécifique à la mise en place d'une connexion SSH se trouve **ici**.*

L'installation du chainage de clés par *keychain* permettra à l'utilisateur *user_a* de la machine cliente « machine A » de se connecter sur la « machine B » en tant que *user_b* sans entrer de mot de passe ou de *passphrase*.

L'intérêt principal est d'éviter d'entrer un mot de passe ou une *passphrase* à chaque connexion SSH mais seulement une fois, lors du lancement de votre « machine A », et pas du tout pour vos shells en tâches de fond.

Une application possible est de pouvoir utiliser les ressources du serveur SSH au travers de scripts shell. Par exemple, les scripts de sauvegarde (avec les commandes scp ou rsync utilisant elles-mêmes le protocole SSH) d'une « machine A » vers une « machine B » pourront alors être lancés en tâches de fond (cronjob) sans avoir à se soucier de quoi que ce soit. Presque le bonheur ! On verra comment réaliser cela dans un exemple en fin de ce tutoriel.

Attention cependant à conserver une configuration sécurisée, notamment si vous êtes plusieurs utilisateurs sur la même machine cliente. En particulier, l'utilisateur root ne doit jamais disposer de ce mode de connexion « automatisée ».

Tout le tutoriel est réalisé en ligne de commande mais ces manipulations sont également possibles en mode graphique.

Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

II - Prérequis

L'utilisateur *root* est requis pour la plupart des installations ci-dessous, même si je n'évoque ou n'explicite pas l'utilisation de « *sudo* », à envisager dans le cas des distributions *Ubuntu*. Les utilisateurs *user_a* et *user_b* devront exister sur chacune des deux machines, disposer de leurs répertoires HOME (*man adduser*) respectifs, et idéalement être autorisés à exécuter du script bash (vous pouvez vérifier que c'est le cas dans */etc/passwd*).

Exemple sur la machine A

```
sudo adduser user_a
cat /etc/passwd
#... d'autres utilisateurs listés
user_a:x:1003:1003:,,,:/home/user_a:/bin/bash
```

II-A - Installer les paquets nécessaires

Machine A : ligne de commande en tant que root

```
aptitude install ssh openssh-client keychain
```

Machine B : ligne de commande en tant que root

```
aptitude install ssh openssh-server
```

1. S'il s'agit d'une réinstallation, pensez bien à stopper tous les processus SSH et « purger » l'installation existante avant de procéder à la réinstallation (*man aptitude*),
2. *aptitude* a disparu des installations standards depuis la version *ubuntu 10.10*. Il faut donc l'installer si vous êtes dans ce cas (avec *sudo apt-get* pour les "*ubuntistes*").

II-B - Configurer votre client et serveur SSH

Le but ici n'est pas tant de vous expliquer le détail de la configuration d'un client et d'un serveur SSH que d'attirer votre attention sur quelques aspects minimalistes de sécurité nécessaires et préalables aux exemples et explications qui suivront.

Les fichiers de configuration concernés :

Machine A : fichier de configuration côté client SSH

```
/etc/ssh/ssh_config
```

Machine B : fichier de configuration côté serveur du daemon (ou service) SSH

```
/etc/ssh/sshd_config
```

- Le chemin peut différer quelque peu selon votre distribution. Une petite recherche en ligne de commande (*find /etc -type f -name ssh*_*) vous orientera sur le chemin de ces fichiers.
- Après modification de la configuration du daemon SSH (*sshd_config*), comme indiqué **ici**, il faut redémarrer le service pour prendre en compte les changements de configuration.

II-B-1 - Changer le port d'écoute du serveur SSH

Il est vivement recommandé de changer le port d'échange SSH tant les tentatives de connexion sur le port SSH standard sont fréquentes par les hackeurs

Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

du dimanche : pas bien violent mais énervant, et potentiellement destructeur. Ceci est d'autant plus applicable si vous n'avez pas un pare-feu robuste, voire pas de pare-feu du tout (hormis celui de la box de votre FAI par exemple). Auquel cas vous devriez envisager au minimum l'installation et la configuration de *iptables* et *fail2ban*, ou similaires selon votre distribution.

Pour les exemples et explications ci-dessous, nous utiliserons donc le port d'écoute 222 au lieu du port SSH par défaut 22 (mais choisissez de préférence un autre port - parmi les ports valides et disponibles - pour votre propre configuration tant cette « variante » est souvent utilisée) :

machine B : vim /etc/ssh/sshd_config

```
# trouver et modifier la ligne «Port 22» comme suit :
Port 222
```

Editeur
La modification est à effectuer en étant connecté en root. J'utilise ici vim comme éditeur de texte en ligne de commande (dans un terminal) mais vi, nano ou même gedit (en mode graphique) font tout aussi bien l'affaire selon vos habitudes ou préférences.

Autres ports utilisables
Vous ne pouvez pas choisir n'importe quel port en alternative au port 22. Contrôlez les ports réservés (cat/etc/services) ou ceux déjà utilisés avec netstat -tulp sur votre machine.

II-B-2 - Utiliser le protocole SSH version 2

Vous verrez encore la possibilité d'utiliser le protocole SSH en version 1 dans vos fichiers de configuration du serveur SSH pour des raisons de compatibilité ascendante, mais il est vivement recommandé de ne plus utiliser ce protocole au profit du seul protocole SSH en version 2, car il a des défauts de conception qui facilitent le crack de mots de passe. Pour les curieux (-ses), les différences entre les deux protocoles sont décrites **ici**.

machine B : vim /etc/ssh/sshd_config

```
# s'assurer que cette ligne ne soit pas commentée
# et supprimer le «protocole 1» s'il apparaît en seconde option
# tel que dans cette ligne : Protocol 2,1
Protocol 2
```

II-B-3 - Interdire la connexion SSH en tant qu'utilisateur root

Comme bien décrit dans **ce tutoriel** que je me permets de répéter quasiment mot pour mot pour l'occasion :

« Les scripts automatisés (notamment) qui tentent de se connecter par SSH sur vos serveurs utilisent principalement les logins couramment utilisés dont [l'utilisateur] root fait bien sûr partie. Le fait de laisser l'accès à [l'utilisateur] root par [une connexion] SSH fait gagner une étape à un pirate qui souhaite s'introduire dans votre système. En effet, le fait de bloquer l'accès à [l'utilisateur] root entraîne la nécessité à un pirate de trouver un nom d'utilisateur autre, existant sur votre système. Sans cela, le pirate dispose donc du nom d'utilisateur, il ne lui reste plus qu'à trouver le mot de passe de [l'utilisateur] root pour s'introduire [par effraction dans votre système]. »

machine B : vim /etc/ssh/sshd_config

```
# S'assurer que l'option est décommentée et réglée sur «No»
PermitRootLogin no
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

N'oubliez pas de relancer le daemon (en français : démon, ou « service ») ssh afin que ces modifications soient prises en compte.

machine B : relancer le daemon ssh en tant que root

```
/etc/init.d/ssh stop
/etc/init.d/ssh start
# ou en alternative : /etc/init.d/ssh reload
```

Vérifions que le port d'écoute est bien actif en ipv4 (et en ipv6 si actif)

machine B : vérifier le port d'écoute SSH

```
netstat -na | grep 222
tcp        0      0 0.0.0.0:222           0.0.0.0:*               LISTEN
tcp6       0      0 :::222                :::*                    LISTEN
```

II-B-4 - Autres recommandations de sécurité

Cela peut paraître évident, mais il est inutile d'utiliser une connexion sécurisée en SSH si vous avez des trous béants de sécurité par ailleurs : il s'agit donc de s'assurer que ces failles éventuelles disparaissent de votre configuration, présentes par défaut sur une distribution *debian lenny* par exemple.

Nous supposerons que *nmap* n'est pas installé et nous supprimerons les services telnet, portmap et inetd, et nous interdirons l'accès au compilateur éventuel (gcc), aux installateurs (apt-get, aptitude et dpkg) à tout autre utilisateur que *root* par un petit script shell :

Machine B : connecté en root dans un terminal, exécutez ce script

```
#!/bin/sh
# SERVICES «INUTILES»
echo -n "Removing and shutting down unsecure services: "
[[ $(dpkg -l | grep portmap | cut -d" " -f1) = "ii" ]] && aptitude remove portmap && update-rc.d -f
 portmap remove
update-rc.d -f inetd remove 1>/dev/null
[[ $(dpkg -l | grep ppp | cut -d" " -f1) = "ii" ]] && aptitude remove ppp
[[ $(dpkg -l | grep telnet | cut -d" " -f1) = "ii" ]] && aptitude remove telnet
echo "done"

# PERMISSIONS du compilateur et installateur
echo -n "Removing compile and install rights to others: "
chmod o-x /usr/bin/gcc-4.3
chmod o-x /usr/bin/make
chmod o-x /usr/bin/apt-get
chmod o-x /usr/bin/aptitude
chmod o-x /usr/bin/dpkg
echo "done"
```

- *portmap* est utilisé notamment pour les services RPC (Remote Procedure Call) à bannir sur un serveur sécurisé. Ce qui sous-entend aussi que vous n'avez pas installé d'interface graphique (Gnome). Si ce n'est pas le cas, ou que vous avez besoin de ce service, il reste cependant possible de le conserver et de gérer l'aspect sécurité au travers du réglage de votre pare-feu (quelle qu'en soit la solution : *iptables* sur votre serveur, ou une machine amont dédiée),
- *ppp* est le service du protocole "Point to Point Protocol" à bannir sur un serveur sécurisé,
- *telnet* est l'ancêtre peu sécurisé de SSH,
- *inetd* (Internet Super Server) est l'ancêtre peu sécurisé de Xinetd, un service (daemon) Unix qui permet de gérer les connexions à des services réseau.

Insérer ce bout de script dans votre script principal de mise en production d'un serveur peut s'avérer utile. Bien sûr, n'oubliez pas de rendre le script lisible et exécutable uniquement pour l'utilisateur root ou votre compte administrateur.

Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Ces quelques recommandations sont nécessaires mais loin d'être suffisantes pour sécuriser un serveur.
Entre autres, il convient de rappeler trois actions indispensables :

- vérifier la robustesse des mots de passe (cf la note sur **john the ripper**) et les changer régulièrement ;
- mettre régulièrement son système à jour (*aptitude update* et *aptitude safe-upgrade*) en ayant bien sûr référencé les dépôts (http://security.debian.org/) dans votre liste de sources ;
- protéger son système, et notamment régler correctement son pare-feu : limiter l'accès aux ports strictement nécessaires et "drop" de l'icmp.

Pour les distributions *debian*, voir également **ces recommandations supplémentaires**.

Enfin, pour clore ce point, une bonne action complémentaire peut être d'installer et configurer **fail2ban** en combinaison avec votre pare-feu et votre serveur.

Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

III - Installer et configurer le chainage des clés

Bon, on y est maintenant. Comme évoqué, le chainage des clés par *keychain* permettra une connexion SSH sans entrer de mot de passe (*passphrase*), sauf en cas de redémarrage de la machine cliente.

III-A - Générer un jeu de clés publique et privée

On utilisera *ssh-keygen* afin de générer un jeu de clés sur la machine cliente, qui pourra ainsi échanger sa clé publique avec la machine serveur et être autorisée à se connecter sur celle-ci.

Deux algorithmes de chiffrement sont disponibles avec l'outil *ssh-keygen* que nous utiliserons :

- RSA : Rivest-Shamir-Adleman ;
- DSA : Digital Signature Algorithm.

À taille identique de clés, la difficulté pour casser les clés est la même pour les deux algorithmes.
Il y a deux différences principales entre ces algorithmes :

a l'algorithme DSA n'assure que la signature alors que RSA réalise à la fois la signature et le chiffrement (souvent réalisé par négligence ou simplification dans un fichier unique), ce qui peut être interprété comme une faiblesse, car le protocole SSH n'utilise la clé que pour la signature. Or il est préférable de limiter la quantité d'informations concentrée dans un fichier, notamment s'il est chiffré. On peut considérer cela comme un avantage au chiffrement DSA ;

b la taille de clé de l'algorithme DSA est figée à 1024 bits alors que RSA permet un chiffrement par défaut à 2048 bits (sauf si l'argument *-b* de la commande *ssh-keygen* est précisé avec une autre valeur - de 768 à 4096 bits). On pourrait considérer cela comme un avantage au chiffrement RSA.

En conclusion, les différences sont minces et le choix est donc dans l'épaisseur du trait. Il doit plutôt porter sur la taille de la clé servant à la signature. 1024 bits sont a priori suffisants, donc nous utiliserons ici la clé DSA. Utilisez l'algorithme RSA si vous souhaitez chiffrer sur 2048 ou 4096 bits.

On quitte l'utilisateur root (ou les commandes sudo de votre compte administrateur) et on se connecte avec le compte de l'utilisateur user_a de la machine cliente.

Pour la *passphrase* :

- définissez et **notez** votre *passphrase* à ce stade ;
- il est recommandé de tester la robustesse de votre *passphrase* avec l'utilitaire **John the ripper** (cf exemple ci-dessous) ;
- il faut toujours entrer une *passphrase* même si *ssh-keygen* permet de ne pas le faire ;
- la robustesse de la *passphrase* est testée par *ssh-keygen*. Vérifiez de ne pas avoir reçu de message d'erreur (et donc que la clé a bien été générée).

Machine A : tester la robustesse des mots de passe

```
# installer l'utilitaire
aptitude install john

# lancer l'utilitaire sur le fichier des mots de passe
john /etc/shadow
```

Machine A : connecté en tant que user_a dans un terminal

```
# on se connecte en tant que user_a
su user_a
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Machine A : connecté en tant que user_a dans un terminal

```
# on se place dans le répertoire HOME de user_a
cd ~

# on génère le jeu de clés (publique et privée):
ssh-keygen -t dsa

# qui demande la saisie d'une passphrase :
Generating public/private dsa key pair.
Enter file in which to save the key (/home/user_a/.ssh/id_dsa):
Created directory '/home/user_a/.ssh'.
Enter passphrase (empty for no passphrase): doit être robuste
Enter same passphrase again:
Your identification has been saved in /home/user_a/.ssh/id_dsa.
Your public key has been saved in /home/user_a/.ssh/id_dsa.pub.
The key fingerprint is:
54:3e:fc:3d:bd:b3:55:bb:2e:ed:78:ed:4c:b3:6c:6e user_a@machineA
The key's randomart image is:
+--[ DSA 1024]----+
|          .      |
|         +       |
|        . +      |
|       .   o . . |
|        S   . o o|
|               .+|
|              .=+|
|             .oEO|
|             .OXo|
+-----------------+
```

Vérification des permissions sur le jeu de clés :
ls -lstha ./.ssh/
4,0K -rw------- 1 user_a user_a 736 2011-05-21 09:12 id_dsa_dvp
4,0K -rw-r--r-- 1 user_a user_a 612 2011-05-21 09:12 id_dsa_dvp.pub
Ce qui signifie :

- accès à la clé privée en lecture/écriture uniquement pour l'utilisateur *user_a* (*chmod 600*) ;
- accès à la clé publique en lecture/écriture pour l'utilisateur *user_a* et en lecture seule pour les autres (*chmod 644*).

Vous pouvez changer a posteriori la passphrase en utilisant l'option -p comme suit : ssh-keygen -p
À noter que vous devrez effectuer à nouveau les étapes qui suivent si vous opérez le changement de passphrase et que vous souhaitez automatiser la connexion.

III-B - Autoriser la clé publique de A sur le serveur B

À ce stade, on ne peut pas encore se connecter à la machine B sans entrer de *passphrase*.

Il faut informer le serveur que la clé publique de la machine A est autorisée pour initier une connexion SSH.

Pour cela, il faut copier la clef publique de l'utilisateur *user_a* de la machine cliente dans le fichier ~/.ssh/authorized_keys de la machine B (serveur SSH) à laquelle l'utilisateur *user_a* voudra se connecter depuis sa machine A via le protocole SSH. Tout s'opère donc depuis la machine cliente.

Sur les distributions de type *debian*, on peut utiliser la commande toute faite *ssh-copy-id* :

Machine A : copier la clé publique sur la machine B dans les clés autorisées

```
ssh-copy-id -p222 -i ~/.ssh/id_dsa.pub user_b@ip_machine_B
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

En alternative pour les autres distributions, mais également valable pour les *debian* :

Machine A : copier la clé publique sur la machine B dans les clés autorisées

```
# Copier la clé publique de A vers B
scp -P 222 /home/user_a/.ssh/id_dsa.pub user_b@ip_machineB:~/.ssh

# Ajouter le contenu de la clé publique dans le fichier authorized_keys de la machine B
ssh -p 222 user_b@ip_machineB "cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys"

# Supprimer la clé publique (celle de la machine A) sur la machine B
ssh -p 222 user_b@ip_machineB "rm ~/.ssh/id_dsa.pub"
```

Enfin, on vérifie que la clé publique de la machine A ait bien été ajoutée aux clés autorisées sur la machine B :

Machine A : vérifier le résultat

```
# Vérifier en comparant le contenu du fichier authorized_keys de la machine B avec la clé publique de la machine

# afficher le fichier de la machine B (depuis la machine A)
ssh -p 222 user_b@ip_machineB "cat ~/.ssh/authorized_keys"

# afficher la clé de la machine A
cat ~/.ssh/id_dsa.pub
```

III-C - Vérifier et tester la connexion SSH

À ce stade, on a d'ores et déjà utilisé la connexion SSH pour copier la clé publique de l'utilisateur *user_a* vers la machine B. Vous avez donc pu rencontrer quelques problèmes dès cette étape, pour lesquels on va essayer d'anticiper les solutions possibles. Sachant que la plupart du temps, ce type de problème est lié à la configuration de votre client SSH (machine A), et dans une moindre mesure, parfois à votre serveur SSH (machine B).

Pensez à utiliser le mode verbeux de la commande ssh, cela vous aidera à résoudre les problèmes éventuels ou à poser vos questions sur les forums avec des éléments concrets :
ssh -p 222 ***-vvv*** *user_b@machineB*

Pensez à vérifier les points suivants :

- /home/*user_a*/.ssh : permission du répertoire réglé sur *chmod 700* ;
- /home/*user_a*/.ssh/known_hosts : *chmod 644* ;
- /home/*user_a*/.ssh/known_hosts: supprimer la clé publique (RSA) de la machine B ;
- /home/autres_users/.ssh/known_hosts : ne contient pas déjà la clé publique de la machine B que le serveur ssh de la machine B est bien en écoute sur le port que vous adressez (ici 222) : *netstat -na | grep 222*.

Host Key verification failed
C'est vraisemblablement l'erreur la plus commune. Plusieurs raisons peuvent causer cette erreur. Plutôt que de les évoquer de façon exhaustive, je vous propose une solution potentielle, ou tout du moins une configuration minimaliste du client, évitant tout risque de destruction massive : le principe est de créer une configuration client spécifique pour l'utilisateur *user_a* afin d'éviter les erreurs de manipulations et de mettre ainsi en péril votre configuration.

Voilà un exemple de configuration SSH cliente et de son mode d'utilisation :

Machine A : configuration SSH client

```
# copier le fichier de base ssh_config
sudo cp /etc/ssh/ssh_config /home/user_a/.ssh/ssh_config

# Attribuer cette config ssh à l'utilisateur user_a
sudo chown user_a:user_a /home/user_a/.ssh/ssh_config

# Se connecter en tant que user_a
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Machine A : configuration SSH client

```
su user_a

# vider le fichier known_hosts
# (ou supprimer la clé hôte de la machine B si vous avez d'autres
# connexions à maintenir)
> /home/user_a/.ssh/known_hosts

# Editer le fichier de config de l'utilisateur user_a
vim /home/user_a/.ssh/ssh_config

# voilà ce que vous allez conserver :
Host *
CheckHostIP no
StrictHostKeyChecking no
Port 222
Protocol 2
SendEnv LANG LC_*
HashKnownHosts yes
GSSAPIAuthentication yes
GSSAPIDelegateCredentials no

# sortez de l'éditeur en sauvegardant (touche ESC + :wq! )
```

Machine A : utilisation de la configuration spécifique de l'utilisateur user_a

```
# on ajoute l'option -F fichier de config à utiliser par ssh
ssh -p 222 -F /home/user_a/.ssh/ssh_config user_b@ip_machineB

# Si cette option a résolu le problème de connexion, couper la connexion
exit

# et reprenez les étapes précédentes en ajoutant l'option -F
# remarque: notez au passage le "P" majuscule pour le port
scp -P 222 -F /home/user_a/.ssh/ssh_config /home/user_a/.ssh/id_dsa.pub user_b@ip_machineB:~/.ssh

ssh -p 222 -F /home/user_a/.ssh/ssh_config
 user_b@ip_machineB "cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys"

# Supprimer la clé publique (celle de la machine A) sur la machine B
ssh -p 222 -F /home/user_a/.ssh/ssh_config user_b@ip_machineB "rm ~/.ssh/id_dsa.pub"
```

Autres problèmes fréquents :

- oubli de recharger la configuration ssh côté serveur après avoir effectué des modifications (taper */etc/init.d/ssh reload*) ;
- permissions incorrectes sur les répertoires et fichiers SSH : cela peut être le cas si vous n'avez pas utilisé les moyens standards (*ssh-copy-id* par exemple). Auquel cas il faudrait rétablir la situation telle que décrite ci-dessous :

Machine cliente « A » (propriétaire : *user_a*)		**Machine serveur SSH « B » (propriétaire : *user_b*)**	
fichier ou répertoire	**droits**	**fichier ou répertoire**	**droits**
fichier: Clé privée	700	Répertoire /home/*user_b*/.ssh	700
fichier: Clé publique	600	fichier : /home/*user_b*/.ssh/authorized_keys	600
fichier: *known_hosts*	644	Recharger la configuration en cas de modification : */etc/init.d/ssh reload*	-

Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

III-D - Automatiser le processus de connexion par session

Afin d'éviter de ressaisir systématiquement sa *passphrase* à chaque connexion lors la session de l'utilisateur *user_a* de la machine cliente, il faut utiliser *ssh-agent*, qui conservera la clef en mémoire.

Il faudra alors informer l'agent SSH de l'existence de votre clé par la commande *ssh-add*. Quand on lancera la connexion SSH, l'agent SSH communiquera la clé publique au processus SSH afin d'assurer les échanges de clés et donc la connexion.

Le seul bémol (mais qui est aussi un gage de sécurité) est que ce processus d'automatisation est valide uniquement pendant la durée de votre session. Nous verrons dans les étapes suivantes comment le rendre persistant.

Machine A: lancer l'agent SSH

```
eval $(ssh-agent)
Agent pid 29993
```

Machine A: communiquer la clé à l'agent SSH

```
ssh-add ~/.ssh/id_dsa
Enter passphrase for /home/user_a/.ssh/id_dsa: entrez votre passphrase
Identity added: /home/user_a/.ssh/id_dsa
```

C'est évident mais je préfère le rappeler : il faut entrer la même *passphrase* que celle que vous aviez pris le soin de noter lors de sa **création**.

III-E - Une connectivité plus « permanente »

III-E-1 - Pour la ligne de commande

Il y a deux solutions principales pour éviter de recommencer l'ajout de la clé à l'agent SSH à chaque nouvelle ouverture de session :

- intégrer l'agent SSH *ssh-agent* directement dans le profil utilisateur ;
- utiliser l'outil *keychain*.

On ne décrira pas la première solution car elle est très bien expliquée **ici**.

Focalisons notre attention sur l'outil *keychain* disponible dans la majorité des distributions Linux (par exemple, disponible pour Debian, Ubuntu, Gentoo, Mandriva) :

Machine A : lancer keychain à la connexion en ligne de commande

```
# Intégrons le lancement de keychain au profil de l'utilisateur user_a
MON_PROFIL=~/.bashrc
test -f $MON_PROFIL && echo "profil: ok" ||
 echo "Cherchez le fichier de votre profil svp (dépend de votre distribution)"
echo "# SSH machine A vers machine B" >> $MON_PROFIL
echo "/usr/bin/keychain ~/.ssh/id_dsa" >> $MON_PROFIL
echo "source ~/.keychain/$HOSTNAME-sh >/dev/null" >> $MON_PROFIL

# nota 1: si vous utilisez csh ou fish, adapter avec ceci
# source ~/.keychain/$HOSTNAME-csh >/dev/null
# source ~/.keychain/$HOSTNAME-fish >/dev/null
# nota 2: "source" s'écrit aussi avec un point

# il faut se reconnecter pour que le profil soit pris en considération
exit
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Machine A : lancer keychain à la connexion en ligne de commande

```
su user_a
# entrer la passphrase de votre clé une dernière fois
```

La *passphrase* de la clé d'identification de la machine A et le mot de passe de l'utilisateur *user_b* ne vous seront plus demandés. Seule la *passphrase* vous sera demandée, et ceci uniquement dans le cas du redémarrage de la machine A.

III-E-2 - Pour vos scripts bash

Aussi bien pour vos scripts lancés manuellement que ceux lancés en tâche de fond (cronjobs), il faut lancer l'agent SSH. Pour cela, il suffit d'ajouter quelques lignes en début de script (bash) :

Machine A : script bash

```
#!/bin/bash
# mon script de cron

KEYCHAIN_INSTALLED=$(aptitude search keychain | egrep "^i")
[[ ! -z $KEYCHAIN_INSTALLED ]] && [[ -f ~/.keychain/$HOSTNAME-sh ]] && source ~/.keychain/$HOSTNAME-sh
[[ ! -f ~/.keychain/$HOSTNAME-sh ]] && (echo "Backup stopped: keychain problem" | tee -a $LOG) &&
 exit 1

# la suite de votre script avec les commandes utilisant le protocole ssh (ssh, scp ou rsync)
# ...
```

- pensez à adapter la source si vous utilisez d'autres interpréteurs (csh, fish) ;
- il faut bien sûr que l'utilisateur exécutant le script soit aussi le possesseur de l'authentification SSH.
  Dit autrement, il faut que le script appartienne à l'utilisateur *user_a* (*man chown*), qu'il soit exécutable (*man chmod*) par *user_a*, et s'il est lancé en tâche de fond (*man crontab*), qu'il se situe bien dans la *crontab* de l'utilisateur concerné (*user_a* dans notre cas).

III-F - Connexions chainées avec les machines A, B, C

Une remarque s'impose sur la connexion SSH d'une machine A vers une machine C en passant par la machine B. Dans ce cas, il faut que le couple des machines B/C soit configuré comme on l'a fait précédemment pour le couple A/B, et utiliser l'option *-A* afin de transmettre l'agent SSH :

Machine A : shell

```
SSH_PORT_B=222
SSH_PORT_C=222
ssh -A -p $SSH_PORT_B user_b@ip_machineB "ssh -p $SSH_PORT_C user_C@ip_machineC"
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

IV - Exemple d'utilisation : sauvegarde distante

On se restreint à la sauvegarde de quelques répertoires choisis d'une machine cliente « A » vers une machine serveur « B » en utilisant *rsync*. Sachant que *rsync* utilise le protocole SSH, on va ainsi pouvoir écrire un petit script de sauvegarde automatique.

IV-A - Configuration

On conserve la même configuration que précédemment (machines A et B avec les utilisateurs *user_a* et *user_b*) et on installe *rsync*.

Machine A : installer rsync

```
aptitude install rsync
```

Machine B : créer un répertoire de backup pour le user_b

```
su user_b
mkdir ~/backup
```

IV-B - Script de sauvegarde (rsync)

On va distinguer deux points : le script lui-même et son automatisation.

IV-B-1 - Créez le script de sauvegarde

Machine A : script shell « /etc/cron.daily/remotebackup »

```
#!/bin/bash
# Remote backup machine A to machine B

# SOURCE
SRC=~

# DESTINATION
DEST="user_b@ip_machineB:~/backup"

# EXCLUDED DIRS
EXCLUDED_DIRS="--exclude-from=/etc/rsync.exclude"

# LOG
LOG="/var/log/rsync.log"

# LOG HEADER
DELIMITER="-----------------------\n"
WHEN=$(date "+%F %H:%M:%S")
WHO=$(whoami)
WHAT=$(echo "$0 (PID $$)")
echo -e $DELIMITER $WHEN $WHO $WHAT >> $LOG

# RSYNC OPTIONS
REMOTE_SSH="-e ssh"
OPTIONS="-avrz --stats --force --delete-excluded"

function pls_wait(){
    local count=$1
    local total=$2
    local progress=$(((100*$count/$total)))
    local char="\x08\x08"
    [[ $progress -lt 10 ]] && char="\x08"
    echo -ne "$char$progress%$char"
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Machine A : script shell « /etc/cron.daily/remotebackup »

```
}

# STARTED
STARTED=$(date "+%s")

# wait for other rsync pid if running
TIMEOUT=300
SLEEP_FREQ=$((TIMEOUT/10))
other_rsync_running=$(ps -ef | grep rsync | grep -v grep)
[[ ! -z $other_rsync_running ]] && \
echo "Other rsync process(es) are running, - pls wait till 100% of $TIMEOUT sec"

while :; do

    other_rsync_running=$(ps -ef | grep rsync | grep -v grep)
    [[ -z $other_rsync_running ]] && echo -e "\nno other rsync running: Ok" && break

    CURRENT=$(date "+%s")
    RUNNING_SINCE=$(( CURRENT - STARTED ))
    [[ "$RUNNING_SINCE" -ge "$TIMEOUT" ]] && \
 echo -e "\nOther rsync's pid running for too long Vs TimeOut $TIMEOUT - process aborted" | tee -
a $LOG && \
 exit 1

    pls_wait $RUNNING_SINCE $TIMEOUT

    sleep $SLEEP_FREQ
done

#SSH KEYCHAIN -for user_a-
KEYCHAIN_INSTALLED=$(aptitude search keychain | egrep "^i")
[[ ! -z $KEYCHAIN_INSTALLED ]] && [[ -f ~/.keychain/$HOSTNAME-sh ]] && source ~/.keychain/$HOSTNAME-sh
[[ ! -f ~/.keychain/$HOSTNAME-sh ]] && (echo "Backup stopped: keychain problem" | tee -a $LOG) &&
 exit 1

# RSYNC GLOBAL
echo "Running /usr/bin/rsync $REMOTE_SSH $OPTIONS $EXCLUDED_DIRS $SRC $DEST" | tee -a $LOG
(/usr/bin/rsync $REMOTE_SSH $OPTIONS $EXCLUDED_DIRS $SRC $DEST >> $LOG) &
RET=$(echo $?)

# WAIT
echo -n "Waiting for the rsync pid's end: " | tee -a $LOG
wait
echo "Ok" | tee -a $LOG
```

Et voilà le fichier contenant les répertoires à exclure de la copie synchronisée. Dans notre cas de sauvegarde du répertoire /home de l'utilisateur *user_a*, c'est inutile, mais il peut être intéressant de voir comment vous pouvez exclure des sous-répertoires de votre répertoire */home*, ou d'autres répertoires, si vous étendez la portée de la sauvegarde :

Machine A : créer un fichier des répertoires à exclure de la synchronisation

```
# Directory or file PATTERNS contained to be excluded from SRC the copy to DEST
# So these are RELATIVE pathes to the SRC directory
# Warning: if SRC does not contain anymore these directories or files, THEY WILL BE DELETED on DEST


# Les répertoires système exclus
/tmp/
/proc/
/sys/
/dev/
/mnt/
/sys/

# divers
/media/
/lost+found/
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Machine A : créer un fichier des répertoires à exclure de la synchronisation

```
# Évidemment auto exclusions
/var/log/rsync.log
/etc/rsync.exclude
```

N'oubliez pas de rendre le script bash exécutable par l'utilisateur *user_a* :

- en notation relative :
  chmod +x /etc/cron.daily/remotebackup
- ou mieux, en notation octale absolue :
  chmod 700 /etc/cron.daily/remotebackup

Évitez les noms exotiques et les extensions pour les noms de fichiers exécutés en tant que tâche automatisée (crontab).
Notamment, les distributions debian n'autorisent pas les points dans les noms de fichiers placés dans /etc/cron.(d|daily|weekly|monthly).

Quelques explications du script :

On ne copie pas un répertoire local vers un autre répertoire distant (*man scp*), mais on lance une copie incrémentale synchronisée (*man rsync*) entre le répertoire source */home* de l'utilisateur *user_a* de la machine A et le répertoire de destination */home/user_b/backup*. À moins d'être utilisée comme un service, la commande *rsync* n'a pas besoin qu'on lui spécifie le port SSH et reprend d'elle-même la configuration (et le protocole) de SSH. On aura vérifié dans le script, préalablement à la synchronisation, que d'autres opérations du même type (d'autres processus *rsync*) ne sont pas en cours d'exécution : si tel est le cas, on attend patiemment qu'elles soient terminées pour ne pas surcharger le CPU et surtout, saturer la bande passante. Si ce délai d'attente *TIMEOUT* est dépassé, on ne lance pas la synchronisation.

IV-B-2 - Automatiser le lancement du script

Passez le script en tâche de fond automatisée pour l'utilisateur *user_a*. Par exemple, on lance une sauvegarde tous les jours à 23h50 par exemple. Pour les distributions de type *debian*, l'utilisation de la table des tâches planifiées (*crontab*) est un peu particulière (l'utilisation est simplifiée) :

Machine A : table des tâches du user_a

```
# on se connecte en tant qu'utilisateur user_a
su user_a

# on édite la table des tâches de fond (la «crontab»)
crontab -e

# voilà ce qu'on va y insérer
SHELL=/bin/bash
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow usercommand
50 23 * * * user_a test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )

# quitter en sauvegardant (dans mon cas, ESC+:wq ! avec l'éditeur vim)

# vérifier en listant les tâches de fond de l'utilisateur user_a
crontab -l
```


Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

V - Résumé

shell

```
# Créer les utilisateurs ssh sur A et B
adduser user_a # sur la machine A
adduser user_b # sur la machine B

# Installer les outils sur la machine A
aptitude install ssh openssh-client keychain john

# Installer les outils sur la machine B
aptitude install ssh openssh-server

# Configurer le client ssh (machine A)
vim /etc/ssh/ssh_config

# Configurer le serveur ssh (machine B)
vim /etc/ssh/sshd_config

# Vérifier la "robustesse" de ses mots de passe
john /etc/shadow

# Créer le jeu de clés sur A
ssh-keygen -t dsa

# Copier et autoriser la clé publique de A sur B
ssh-copy-id -p222 -i ~/.ssh/id_dsa.pub user_b@ip_machine_B

# Lancer l'agent ssh sur A
eval $(ssh-agent)
ssh-add ~/.ssh/id_dsa

# Automatiser la connexion de A vers B dans le profil utilisateur
/usr/bin/keychain ~/.ssh/id_dsa
source ~/.keychain/$HOSTNAME-sh >/dev/null
```

J'espère que vous aurez trouvé ce tutoriel complet, clair et utile.

Merci également à Thibault Cuvelier, Raymond alias ram-0000 et Jonathan Courtois pour les avis techniques et à Claude LELOUP pour la correction orthographique.

Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

VI - Documentation

VI-A - Algorithmes et clés de chiffrement

- **RFC4432 : clés RSA**
- **Explication descriptive de l'algorithme RSA**
- **Recommandations d'un maitre du cryptage**
- **Taille des clés**

VI-B - Installer un serveur SSH

- **Tutoriel developpez.com**
- **Mini-guide ssh debian**

VI-C - Aspects sécuritaires

- **Manuel de sécurisation de Debian**
- **Les pare-feux en détail**
- **Wiki iptables**
- **Exemples détaillés avec iptables**
- **Wiki fail2ban**
- **Désactiver ipv6**

Les sources présentées sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2011 - Philippe Beaucart. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.