

Installer SSH keychain pour éviter d'avoir à saisir ses passphrases

Lorsqu'on a plusieurs clés SSH et que l'on souhaite éviter de saisir sans arrêt sa passphrase, [keychain](#) est la solution. Cet outil permet de gérer les clés SSH (et GPG mais ce n'est pas le sujet) de façon très pratique et sécurisée. Il agit en tant que frontal aux commandes `ssh-agent` et `ssh-add`. Contrairement à `ssh-agent` qui fonctionne sur une session (du login à la déconnexion), `keychain` conserve un agent fonctionnel tant que le serveur n'a pas été redémarré. Ceci diminue le nombre de fois où vous devrez saisir votre passphrase. De plus, il est facile d'utiliser l'agent dans un script exécuté par cron.

Installation

On utilise le gestionnaire de paquets sous Linux, et sous MacOS on utilisera [Homebrew](#).

```
# sur un système Linux Debian
apt-get install keychain

# sur un MacOS
brew install keychain
```

Ensuite, on édite notre fichier `~/.bash_profile` pour y ajouter le contenu suivant.

```
eval `keychain --eval --agents ssh ~/.ssh/neptune_key ~/.ssh/ma_cle_perso`
```

On indique ici que l'on utilise la commande `eval` via l'option `--eval`, ceci permet de ne pas se soucier du SHELL que l'on exécute. On n'utilise que la partie SSH de l'agent `--agents ssh` et on indique les clés à déverrouiller en argument.

Si vous ne savez pas comment utiliser les clés SSH, vous pouvez aller faire un tour sur l'article [créer vos clés publique et privée SSH](#).

Première ouverture du terminal

La première fois que j'ouvre mon terminal, keychain sait qu'aucune clé SSH n'est chargée, alors il demande les passphrases de chacune des clés. Si deux clés ont la même passphrase, à priori, keychain ne demande la passphrase qu'une seule fois, c'est mon cas dans cet exemple.

```
* keychain 2.8.3 ~ http://www.funtoo.org
* Starting ssh-agent...
* Adding 2 ssh key(s): /Users/matthieu/.ssh/neptune_key /Users/matthieu/.ssh/ma_cle_perso
Enter passphrase for /Users/matthieu/.ssh/neptune_key:
* ssh-add: Identities added: /Users/matthieu/.ssh/neptune_key
/Users/matthieu/.ssh/ma_cle_perso
```

Sessions de terminal suivantes

Lorsque j'ouvre d'autres sessions de mon terminal, keychain indique que des clés sont chargées, il ne redemande donc pas les passphrases.

```
* keychain 2.8.3 ~ http://www.funtoo.org
* Found existing ssh-agent: 33235
* Known ssh key: /Users/matthieu/.ssh/neptune_key
* Known ssh key: /Users/matthieu/.ssh/ma_cle_perso
```

Dans le cas où l'agent est exécuté sur un serveur, la passphrase ne sera demandé qu'après un reboot.

Se connecter à un serveur SSH

La connexion se fait de manière classique, on spécifie simplement la clé à utiliser. La passphrase ne doit pas être demandée si la clé privée correspondante est chargée en mémoire.

```
ssh -i ~/.ssh/neptune_key matthieu@monserveur.local
```

Lister les clés chargées

Cette commande liste les signatures des clés chargées en mémoire.

```
keychain --list
SHA256:zz7Jx7tHoUpCdAAMSRXU7nl+zmbYdKe09oYmQfsUg2s
SHA256:PYdgMViJ70rcL8bVBV0aAsJkglFS7w9xDazNAMxWd84
```

Utiliser depuis un script via cron

Si j'ai un script qui est exécuté par cron (le crontab de ma session), et que ce script doit se connecter à un serveur distant, pour faire une sauvegarde par exemple, voici comment utiliser keychain dans le script.

```
#!/bin/bash

# chemin vers la clé privée à utiliser
SSH_PRIVATE_KEY=~/.ssh/ma_cle_perso

# l'empreinte de la clé
SSH_KEY_FINGERPRINT=`ssh-keygen -E sha256 -lf "$SSH_PRIVATE_KEY" | cut -d\  -f2`

# chargement de keychain en mode silencieux
eval `keychain --quiet --agents ssh --noask --eval "$SSH_PRIVATE_KEY"`

# vérification que la clé privée est bien chargée dans keychain
if keychain --list | grep $SSH_KEY_FINGERPRINT > /dev/null; then

    # on déroule notre script ici.

    # exemple, on veut lister le répertoire distant de monserveur.local
    LS=`ssh -i "$SSH_PRIVATE_KEY" matthieu@monserveur.local ls`
else
    printf "Erreur, clé %s non chargée...\n" "$SSH_PRIVATE_KEY" >&2

    # envoyer un email pour prévenir l'utilisateur, par exemple
fi
```

L'option `--noask` indique à keychain qu'il ne doit pas demander la passphrase, étant donné qu'il s'agit d'un script exécuté sans intervention humaine, personne ne sera là pour la saisir si la clé privée est verouillée. On

teste ensuite que notre clé est bien chargée en mémoire, pour cela on calcule son empreinte, puis on vérifie si elle est dans la liste des clés chargées.

Décharger les clés en mémoire

Si on doit laisser quelqu'un utiliser son ordinateur quelques minutes, on ne veut pas que cette personne puisse se connecter à un de nos serveurs. On doit donc décharger les clés en mémoire.

```
keychain --clear
```

```
* keychain 2.8.3 ~ http://www.funtoo.org
* Found existing ssh-agent: 33235
* ssh-agent: All identities removed.
```

Sur un serveur distant, afin d'améliorer la sécurité de keychain, on peut utiliser cette option dans le `.bash_profile`. Ainsi, on prend pour hypothèse que chaque nouvelle session SSH est considérée comme étant initiée par un intrus, l'intrus doit donc prouver qu'il est un utilisateur légitime en saisissant la passphrase, même si la clé était déjà en mémoire auparavant. Les scripts lancés par cron pourront toujours s'exécuter lorsque vous vous déconnecterez, sauf si un intrus s'est réellement connecté, et dans ce cas, la passphrase n'aura pas été saisie.

Sur un serveur, il est courant de mettre cette ligne dans le fichier `~/.bash_profile`.

```
eval `keychain --eval --clear --agents ssh ~/.ssh/neptune_key ~/.ssh/ma_cle_perso`
```

Un point sur la sécurité

Utiliser un agent SSH a une incidence sur la sécurité, voici ce qu'il faut savoir.

- Ne copiez pas votre clé privée sur un serveur non administré par vous, si vous le faites, vous partagez votre clé avec l'administrateur, la passphrase protège normalement votre clé, mais si vous avez copié votre clé privée, c'est que vous l'avez sans doute utilisée, vous avez donc partagé votre passphrase avec l'administrateur, même si cette opération n'est pas simple à réaliser, on peut se demander si la clé privée est compromise, à partir de cet instant. Ne stockez vos clés privées que sur des machines dites de confiance, encore mieux, ne lancez vos sessions SSH que depuis votre ordinateur personnel.
- N'utilisez l'agent SSH ou l'agent forwarding que sur des machines administrées par vous car ces deux méthodes créent des variables d'environnement `SSH_AUTH_SOCK` contenant une socket vers l'agent SSH, il suffit à l'administrateur de mettre le chemin vers cette socket dans la variable d'environnement `SSH_AUTH_SOCK` de sa propre session pour avoir accès à toutes vos machines, sans mot de passe.

Concernant ce dernier point, on pourra ajouter ceci dans notre fichier `~/.ssh/config` pour interdire l'agent forwarding.