Table des matières

Les Clés GPG avec SSH 2
Explication 2
Configurer GPG 2
gpg-agent.conf 2
gpg.conf 3
sshcontrol 3
Configurer SSH 4
.bashrc 4
Vérification 4
Exporter sa Clé 5
Connexion 5
Serveur Mode 5

Les Clés GPG avec SSH

Explication

Le but sera d'utiliser GPG (l'agent GPG) en lieux et place de l'agent SSH pour établir une connexion basée sur l'échange de clé privé/public GPG (PGP open source) afin de renforcer la sécurité des connexions distantes entre deux hôtes linux.

Configurer GPG

Dans un terminal, depuis votre profile utilisateur courant.

```
pwd
/home/ma-session
```

gpg-agent.conf

Ajouter ces lignes dans le fichier si elles n'y sont pas, dans certains cas créer le fichier et ajouter y ces lignes:

```
nano .gnupg/gpg-agent.conf
enable-ssh-support
default-cache-ttl 300
max-cache-ttl 1200
```

CTRL+o pour enregistrer puis CTRL+x pour quitter.

```
gpgconf –kill gpg-agent
eval $(gpg-agent –daemon)
```

```
gpg-agent[70012]: gpg-agent running and available
```

Parfait !!

gpg.conf

```
nano.gnupg/gpg.conf
use-agent
```

Idem, ajouter et ou créer avec cette ligne le fichier

sshcontrol

Dans ce fichier nous renseignerons la ou les “keygrip” de clé GPG à utiliser de type **[A]** pour établir une connexion SSH vers un hôte distant.

```
nano .gnupg/sshcontrol
# List of allowed ssh keys.  Only keys present in this file are used
# in the SSH protocol.  The ssh-add tool may add new entries to this
# file to enable them; you may also add them manually.  Comment
# lines, like this one, as well as empty lines are ignored.  Lines do
# have a certain length limit but this is not serious limitation as
# the format of the entries is fixed and checked by gpg-agent. A
# non-comment line starts with optional white spaces, followed by the
# keygrip of the key given as 40 hex digits, optionally followed by a
# caching TTL in seconds, and another optional field for arbitrary
# flags.   Prepend the keygrip with an '!' mark to disable it.
D81XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Pour lister ces Clé GPG:

```
gpg --list-public-keys  --with-keygrip
```

```
gpg --list-public-keys  --with-keygrip
/home/sylvain/.gnupg/pubring.kbx
--------------------------------
pub   rsa4096 2022-04-18 [C] [expire : 2027-04-17]
      B327XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      Keygrip = 3AD06XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
uid           [  ultime ] Sylvain Key (Ma clé pour certifier)
<svalldaura@me.com>
sub   rsa4096 2022-04-18 [S] [expire : 2027-04-17]
      Keygrip = 067EXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
sub   rsa4096 2022-04-18 [A] [expire : 2027-04-17]
      Keygrip = D81XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
sub   rsa4096 2022-04-18 [E] [expire : 2027-04-17]
      Keygrip = C74BXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

sub rsa4096 2022-04-18 [A] [expire : 2027-04-17]
Keygrip = D81XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Vous noterez que le *keygrip* de la sous clé dédié à l’authentification [A] est bien le même

que celui présent dans le fichier *sshcontrol*.

Configurer SSH

Ajouter cette ligne dans le fichier de config SSH.
Ce qui permettra à l'agent GPG de prendre place à celui de SSH. Depuis Ubuntu 21.04 le fichier se situe ici.

```
nano .ssh/config
IdentityAgent /run/user/1000/gnupg/S.gpg-agent.ssh
```

ou bien

```
echo "IdentityAgent /run/user/1000/gnupg/S.gpg-agent.ssh" >> ~/.ssh/config
```

.bashrc

Editer votre profil bash (.bashrc) et ajouter y à la fin cette ligne.
Cela permettra au lancement de votre terminal de charger l'agent GPG.
Vous pouvez aussi ajouter l'entrer dans CRON, mais je trouve que ce n'est pas sa place !

```
#GPG-AGENT LOAD CONF
SSH_AUTH_SOCK=/run/user/1000/gnupg/S.gpg-agent.ssh
```

Noter que pour .zshrc l'ajout de ligne reste la même 

Vérification

Nous allons vérifier que tout soit ok !

```
gpg-agent
gpg-agent[148469]: gpg-agent running and available
```

on dit à l'agent qu'il faut parler sur le TTY courant (pour nous demander le code PIN) :

```
gpg-connect-agent updatestartuptty /bye
```

```
ssh-add -L
```

doit afficher l'empreinte RSA 4096 de la clé
Ensuite mettre la clé sur une machine distante, dans

```
~/.ssh/authorized_keys
```

et s'y connecter :

Exporter sa Clé

```
gpg --export-ssh-key B327XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX >
~/.ssh/yubikey.pub
```

Vous pouvez également modifier votre fichier de configuration ~/.ssh/config pour utiliser la clé publique et l'agent GPG se chargera du reste ...

```
Host serveur
    Hostname 192.168.0.1
    User root
    IdentityFile ~/.ssh/yubikey.pub
```

Connexion

```
ssh user@ip-pc-distant
```

Une fenêtre s'ouvre et vous demande la parphrase de votre clé GPG (si elle en possède une) ou le code **PIN** de votre clé **Yubikey** (si vos clé y sont stocker dedans) pour matcher avec la clé SEC (privé) afin d'établir la connexion.

Serveur Mode

Configurer gpg-agent pour ne pas demander le code PIN

Note: ces étapes ne sont pas nécessaires si vous souhaitez utiliser votre clé Yubikey sur un poste de travail. Elle ne servent que pour une utilisation « sans présence humaine » des clés.

L'avant dernière étape consiste à dire à gpg-agent que l'on ne souhaite pas avoir à saisir le code PIN pour pouvoir utiliser la clé : cela nécessite en effet de pouvoir lancer nos sauvegardes sans la présence d'un humain pour saisir le code PIN !

Pour cela, on va demander à GPG-Agent, via la commande gpg-connect-agent, de se souvenir « en RAM » du code PIN de la carte.

Notez que cela doit donc être fait à chaque démarrage du serveur de sauvegarde. (et remplacez 123456 par votre code PIN bien entendu, et le D27blabla000 par le numéro de série de votre Clé Yubikey, voir avec la commande gpg –edit-card)

```
$ cat >/tmp/pin
123456
<ctrl-d>
$ gpg-connect-agent
> OPTION pinentry-mode=loopback
> /definqfile PASSPHRASE /tmp/pin
> SCD CHECKPIN D276000XXXXXXXXXXXXXXXXXXXXXX
> /bye
$ rm -f /tmp/pin
```

Configurer ses crons pour utiliser l'agent GPG qui tourne

Dernière étape : il faut modifier nos tâches planifiées de sauvegarde pour qu'elle utilisent l'agent GPG (en fait un agent SSH) qui tourne comme source de signature pour l'identification sur les serveurs distants.

pour cela, on ajoute la variable d'environnement suivante aux crontab :

```
SSH_AUTH_SOCK=/run/user/1000/gnupg/S.gpg-agent.ssh
```

Ainsi, SSH utilisera l'agent GPG, qui en RAM a le code PIN pour accéder à la clé et demander la signature de session de connexion SSH, dont les clés privées resteront à l'intérieur de la clé, et donc non exfiltrables.

Si vous souhaiter générer et ajouter vos clé GPG dans une Yubikey:
Les Clés Sécurité Multi-Protocoles