Signer ses commits Git et transférer son gpg-agent sur un serveur distant

GPG, c'est bien. C'est encore ce qu'on a trouvé de mieux pour que tout un chacun puisse s'assurer de l'authenticité d'un message (non-modification de celui-ci et que son auteur est bien celui annoncé) et chiffrer ses messages.

OK, c'est pas un exemple d'ergonomie, OK, c'est pas ma mère qui va s'en servir sciemment tous les jours (sous le manteau, si, puisque les paquets de sa Debian sont signés avec GPG). Mais d'un autre côté, ma mère ne risque pas non plus (et je dirais même encore moins) de se payer un certificat X509 pour signer ses mails. Ne parlons pas de mon père, à côté de lui, ma mère fait figure de hax0r.

Bon, d'accord, c'est un bon gros truc de geek. OSEF, c'est cool quand même, c'était pour dire que c'était robuste et que tout le monde peut l'utiliser sans bourse délier.

Que vous le sachiez (*dans la colle*) ou pas, on peut signer ses commits git avec GPG, histoire d'ajouter encore une couche de sécurité aux modifications qu'on apporte à un logiciel. Des forges comme Gitlab permettent d'ajouter une clé GPG à son profil permettant ainsi de vérifier les signatures des commits d'un projet. Voyez sur ce commit le joli petit bouton « *Verified* ».

Ceci n'est pas un cours sur GPG, on va donc considérer que vous avez déjà une clé GPG.

Signer ses commits Git

Rien de plus simple. Il faut tout d'abord déclarer à Git quelle clé doit être utilisée pour signer ses commits (changez l'empreinte, ça c'est la mienne) :

```
git config --global user.signingkey EA868E12D0257E3C
```

Maintenant, soit vous ajoutez `-S` quand vous committez :

```
git commit -S
```

Soit vous configurez Git pour signer tous vos commits :

```
git config --global commit.gpgsign true
```

Voilà, c'est bon. Pour vérifier un commit :

```
git verify-commit cce09ca
```

C'est bien beau, mais ça m'arrive de développer directement sur des serveurs, et surtout, je développe généralement dans une machine virtuelle sur mon PC. Je ne vais certainement pas aller copier ma clé privée sur les-dits serveurs ou dans la machine virtuelle ! C'est là qu'intervient le transfert du gpg-agent sur le serveur distant.

Ceci n'est toujours pas un cours sur GPG, on va donc considérer que vous avez déjà un gpg-agent fonctionnel sur votre ordinateur.

Transférer son gpg-agent sur un serveur distant

ATTENTION On ne transfère son agent gpg que sur une machine dans laquelle on a confiance, et dont on sait que les personnes y ayant accès ne s'amuseront pas à utiliser votre agent (rien de plus simple si on a un accès root à la machine). Cela vaut aussi pour l'agent ssh !

Premièrement, on va dire à l'agent de créer un *socket* supplémentaire en mettant dans `~/.gnupg/gpg-agent.conf` (remplacez `<user>` par votre login) :

```
extra-socket /home/<user>/.gnupg/S.gpg-agent.extra
```

Ce *socket* a des restrictions que n'a pas le *socket* habituel (ne me demandez pas lesquelles) mais surtout, le logiciel qui vous demandera votre mot de passe (*pinentry* de son petit nom générique) vous présentera la demande de mot de passe différemment d'habitude. Moi, il m'a dit en gros « Cette demande provient d'une machine distante », ce qui permet de repérer d'où vient la demande (déjà pas de votre machine pour déchiffrer un mail par exemple) et de réfléchir à si c'est bien vous qui avez fait une action demandant la clé GPG.

On redémarre l'agent pour prendre en compte la nouvelle configuration :

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

Ensuite, il va falloir modifier sa configuration SSH. Comme un bon adminSys est fainéant, vous avez bien sûr utilisé concierge pour gérer votre fichier `~/.ssh/config`. Il suffit d'ajouter dans le bloc de configuration du serveur souhaité la ligne (remplacez `uid` par votre *uid* (`id` pour le connaître) et `<user>` par votre login):

```
RemoteForward /run/user/<uid>/gnupg/S.gpg-agent /home/<user>/.gnupg/S.gpg-agent.extra
```

Enfin, il faut ajouter ceci dans le `/etc/ssh/sshd_config` du serveur distant (et redémarrer son dæmon ssh après) :

```
StreamLocalBindUnlink yes
```

C'est fini ! Vous pouvez maintenant utiliser votre clé GPG sur un serveur distant en vous y connectant en SSH, sans copier votre clé sur le serveur

Connectez-vous en SSH et testez avec

```
echo "test" | gpg2 --clearsign
```

Si, lorsque vous tentez de signer un commit sur votre serveur distant, cela échoue, assurez-vous que git utilise bien `gpg2` :

```
git config --global gpg.program gpg2
```

Merci à Thomas Citharel pour avoir mis la signature GPG des commits sur le tapis d'une discussion, ce qui m'a poussé à me pencher sur le transfert de l'agent GPG.

Révision #1
Créé 23 janvier 2020 16:12:19 par Luc
Mis à jour 23 janvier 2020 16:14:08 par Luc