

Trucs et astuces SSH

Matthieu Herrb



Capitoul - 23 juin 2022

<https://homepages.laas.fr/matthieu/talks/2022-06-capitoul-ssh.pdf>



Ce document est sous licence

Creative Commons Paternité - Partage Partage dans les mêmes conditions 4.0 International

Le texte complet de cette licence est disponible à l'adresse :

<http://creativecommons.org/licenses/by-sa/4.0/>

Rappel : clés publiques et chiffrement

SSH utilise plusieurs algorithmes de chiffrement et de type de clés associés :

- une paire de clés asymétriques pour l'authentification des hôtes
→ **HostKeyAlgorithms**
- un protocole d'échange de clés choisit une clé secrète symétrique
→ **KexAlgorithms**
- utilisée par un algorithme de chiffrement de la session
→ **Ciphers**
- les messages sont authentifiés par un MAC (message authentication code)
→ **MACs**
- optionnellement une paire de clés asymétriques pour l'authentification des personnes → **PubkeyAcceptedKeyTypes**

Algorithmes dépréciés

- OpenSSH 8.8 (octobre 2021) a supprimé par défaut le support de clés signées utilisant SHA-1 comme fonction de hashage. En particulier **ssh-rsa**.
- OpenSSH 8.5 (mars 2021) change la gestion des clés hôtes et enregistre automatiquement les clés hôtes avec meilleure sécurité.
- OpenSSH 7.6 (octobre 2017) refuse les bi-clés RSA de longueur < 1024 .
- OpenSSH 7.2 (mars 2016) a supprimé par défaut le support d'algorithmes symétriques en mode CBC
- OpenSSH 7.0 (août 2015) a supprimé par défaut le support de clés utilisant l'algorithme DSA (**ssh-dss**) ainsi que l'algorithme d'échange de clés **diffie-hellman-group1-sha1**.

Les algorithmes supprimés par défaut peuvent être réactivés par des options.
cf <https://www.openssh.com/legacy.html>

<https://access.redhat.com/articles/3642912>

Red Hat applique à partir de RHEL 8 (et aussi dans Fedora ?) des niveaux de sécurité stricts pour la cryptographie.

Les algorithmes désactivés dans ce cadre ne peuvent pas être réactivés par une option dans SSH...

→ il faut conserver un client plus ancien pour accéder à des systèmes obsolètes, ou bien mettre à jour tous les systèmes.

Accepter des algorithmes dépréciés

`$HOME/.ssh/config :`

```
Host somehost.example.org
  KexAlgorithms +diffie-hellman-group1-sha1
  HostkeyAlgorithms +ssh-rsa
  PubkeyAcceptedAlgorithms +ssh-rsa
  Ciphers +aes128-cbc
```

Fichier de configuration - options classiques

```
Host anotherhost.example.com
  User joe
  Port 2222
  PasswordAuthentication No
  PreferredAuthentications publickey
  IdentityFile ~/.ssh/id_ed25519
```

Pour tester/voir les options actives : **ssh -G host.example.com**

Fichier de configuration du client - autres astuces

Multiplexage de connexions : tous les SSH passent dans la même connexion

→ plus rapide,

→ pas de déclenchement d'anti-brute force.

```
Host *.l
  Hostname %haas.fr
  User matthieu
  ControlMaster auto
  ControlPersist 600
  ControlPath ~/.ssh/C-%r@%h:%p
```

Plus d'infos : http://man.openbsd.org/ssh_config

Jump host

Utilisation d'un bastion (rebond) de manière transparente :

```
ssh -J bastion.example.com otherhost.example.com
```

ou bien dans \$HOME/.ssh/config :

```
Host *.example.com  
    ProxyJump bastion.example.com
```

Garder ses connexions actives

- **TCPKeepAlive** *bool* (défaut **yes**) contrôle l'émission de paquets *KeepAlive* au niveau du protocole TCP. Provoque la déconnexion au bout d'environ 2mn si connexion down.
Désactiver si connexion non fiable pour éviter déconnexions sauvages.
- **ServerAliveInterval** *n* envoi un paquet SSH *ServerAlive* toutes les *n* secondes. Si plus de **ServerAliveCountMax** perdus → déconnexion.
(default **0** → désactivé)
Permet de maintenir une session NAT active même si terminal inactif.

Exemple : maintient la connexion pendant 10mn en cas d'inactivité.

```
Host yanh.example.com
    TCPKeepAlive no
    ServerAliveInterval 15
    ServerAliveCountMax 40
```

Certificats SSH

Solution au problème de gestion manuelle des clés hôte et personnes :
(`known_hosts` et `authorized_keys`)

- Une clé spécifique est l'autorité de certification.
- Chaque clé (hôte ou personne) est signée par l'autorité de certification
- `sshd_config` permet de faire confiance à toutes les clés signées

→ mettre en place un mécanisme de signature des clés valides.

NB : il ne s'agit pas de PKI X.509, mais de certificats au format PGP.

<https://berndbausch.medium.com/ssh-certificates-a45bdcdfac39>

Questions ?

