Serveur en mode Headless

Table of Contents

Un serveur Headless est simplement un système d'exploitation installé sur un ordinateur sans moniteur, souris, clavier et autres périphériques. On peut faire d'un serveur un serveur Headless en le connectant à un réseau, pour utiliser des outils de mise en réseau tels que ssh et vnc server.

Authentification par mot de passe

L'authentification par mot de passe nécessite l'activation de **PasswordAuthentication** dans `/etc/ssh/sshd_config`:

```
PasswordAuthentication yes
```

Après avoir effectué cette modification, redémarrer le service SSH en exécutant la commande suivante en tant que root :

```
sudo service ssh restart
```

Lorsque SSH demande un identifiant (ce n'est donc pas un problème de port) et un mot de passe (donc **PasswordAuthentication** est défini sur yes), mais un problème de connexion subsiste cela peut être lié à:

1. **AllowGroups** ou **AllowUsers** dans `sshd_config`, si l'une de ces valeurs est définie, il faut soit rajouter le groupe ou le user, soit mettre le user dans l'un des groupes autorisés avec la commande: `sudo usermod -a -G groupname username`
2. un problème d'autorisation,en effet, la façon dont **sshd** fonctionne par défaut est qu'il tente de lire les clés SSH publiques avant de demander un mot de passe, et si `~/.ssh/authorized_keys` a les mauvaises autorisations, **sshd** ne permettra pas de se connecter. Pour s'assurer que sshd a les droits adpatés, utiliser ceci :`chmod -v a-w,u+w ~/.ssh/authorized_keys` (**-v** permet à chmod d'informer des modifications apportées (le cas échéant), **a-w,u+w** consiste à supprimer les autorisations d'écriture pour tous, puis à redonner les autorisations d'écriture à l'utilisateur.)

Afin d'autoriser l'utilisateur root à se connecter, l'option **PermitRootLogin** doit être défini sur “yes”:
`PermitRootLogin yes`

Pour activer les mots de passe vides, mettre **yes**, l'option **PermitEmptyPasswords** doit être défini sur “yes”(NON RECOMMANDÉ):
`PermitEmptyPasswords yes`

Le backend **UsePAM** doit être défini sur “yes”:
- Lorsqu'on veut utiliser l'authentification PAM, les vérifications de compte et les vérifications de session. L'authentification PAM peut alors être autorisé via **ChallengeResponseAuthentication** [1)] et/ou **PasswordAuthentication**. En fonction de la configuration, l'authentification PAM via **ChallengeResponseAuthentication** peut contourner le paramètre de **PermitRootLogin without-password**.
- Lorsqu'on veut juste les vérifications de compte et les vérifications de session sans **Authentification PAM**, il faut définir **PasswordAuthentication** et **ChallengeResponseAuthentication** sur «no».

Si le backend n'est pas configuré, **ChallengeResponseAuthentication** doit être défini sur “no” afin que ssh n'utilise pas un backend non configuré.

Authentification par clé

L’authentification par clé privée augmente la sécurité d’un serveur au niveau du processus de connexion.

Pour mettre en place ce type d’authentification, il va falloir générer 2 clés : La clé privée et la clé publique. La clé publique n’a pas besoin d’être secrète. C’est une clé qui est capable de reconnaître sa clé privée (cryptographie asymétrique). Cette clé sera donc placée sur le serveur afin qu’elle puisse vérifier la clé privée lorsqu'on veutse connecter.

Pour générer la paire de clé utiliser la commande suivante :

```
ssh-keygen -t rsa -b 2048 -C votre@email.com
```

Par défaut ssh-keygen génère une clé RSA de 2048 bits: 1024 est considéré comme la taille de clé minimale pour RSA mais pour un usage général, 2048 est suffisant

Les deux clés sont générées par défaut dans `~/.ssh/`:

- **id_rsa.pub**: La clé publique
- **id_rsa**: La clé privée

Il faut maintenant envoyer la clé publique sur le serveur et la placer dans le fichier `~/.ssh/authorized_keys`. Ce fichier est utilisé lors de l’authentification pour déterminer si la clé privée de l’utilisateur est conforme à la clé publique inscrite dans le fichier.

Pour envoyer la clé publique dans le fichier `~/.ssh/authorized_keys` on peut:

- utiliser **ssh-copy-id** qui permet d’envoyer en remettant les bons droits d’écriture sur le fichier: `ssh-copy-id -i ~/.ssh/id_rsa.pub <username>@<ipaddress>` (si le port SSH n’est pas sur le port standard (22) `ssh-copy-id -i ~/.ssh/id_rsa.pub -p <num_port>`

<username>@<ipaddress>)
- transférer la clé depuis le serveur avec scp `scp ~/.ssh/id_rsa.pub destHost:~ (enter password)`, sur le client, prendre en compte le clé `cat id_rsa.pub >> .ssh/authorized_keys` puis ajuster les privilèges du fichier `chmod 700 .ssh/authorized_keys`

Côté serveur, il faut maintenant indiquer à ssh que l'on souhaite permettre la connexion par clés dans `/etc/ssh/sshd_config`:

```
RSAAuthentication yes
PubkeyAuthentication yes
StrictModes yes
```

Avec **StrictMode yes**, il faut donner les bons droits au répertoire homedir sinon l'authentification risque d'échouer :

```
chmod go-w ~/
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

Recharger le serveur ssh :

```
service sshd reload
```

1)

ChallengeResponseAuthentication ne fournit pas une sécurité supplémentaire en soi. Le terme **ChallengeResponseAuthentication** est simplement un mot clé de configuration **OpenSSH**; il fait référence à la méthode utilisateur **clavier-interactif** dans le protocole SSH