Configurer une authentification SSH avec certificat

Resources

- http://www.ixany.org/fr/articles/presentation-et-utilisation-des-certificats-ssh/
- https://www.ssh.com/
- http://tech.ciges.net/blog/openssh-with-x509-certificates-how-to/
- https://linux-attitude.fr/post/certificats-x509-pour-ssh
- https://ichi.pro/fr/comment-utiliser-les-certificats-ssh-pour-un-acces-serveur-evolutif-securise-et-plus-transparent-125389551658519

Principes

L'utilisation de certificats permet de garantir l'identité du client et du serveur. OpenSSH permet de gérer des certificat dans un **format spécifique**. Ce ne sont pas des certificats x509.

OpenSSH gère des certificats dans un **format spécifique** qui **ne sont pas des certificats x509** utilisables avec SSL/TLS.

La clé publique des utilisateurs et des serveurs doit être signée par la clé privée d'une autorité de certification spécifique à OpenSSH.

- **Intérêt pour le client** : la **clé publique de la CA** permet de vérifier le certificat présenté par le serveur. Il n'est pas nécessaire de renseigner le fichier **know_hosts**.
- **Intérêt pour le serveur** : la **clé publique du CA** permet de vérifier le certificat présenté par le client sans avoir à renseigner le fichier **authorized_keys**.

Configuration du serveur SSH (OpenSSH)

Le fichier de configuration du service SSH est **/etc/ssh/sshd_config**.

Lors de l'installation du serveur SSH ses clés privées et publiques sont générées sous plusieurs formats dans le dossier **/etc/ssh**, pour garder une comptabilité avec des clients plus anciens :

- format **dsa** : **ssh_host_dsa_key** et **ssh_host_dsa_key.pub**
- format **rsa** : **ssh_host_rsa_key** et **ssh_host_rsa_key.pub**
- format **ed25519** : **ssh_host_ed25519_key** et **ssh_host_ed25519_key.pub**
- format **ecdsa** : **ssh_host_ecdsa_key** et **ssh_host_ecdsa_key.pub** ; ce format autorise l'écha,nge de clé ECDH Elliptic Curve Diffie Hellman).

Dans le fichier de configuration du service SSH **/etc/ssh/sshd_config**, il est possible de fixer le type de clé côté serveur avec la directive **HostKey**.

Exemple :

```
HostKey /etc/ssh/ssh_host_rsa_key
```

La vérification de la configuration du service SSH se fait avec la commande suivante :

```
# /usr/sbin/sshd -d
```

Les éléments nécessaires

- la clé privée pour signer les certificats OpenSSH ;
- la clé publique du client et son identité du client ;
- ou la clé publique du serveur pour un certificat serveur et son identité.

Génération de la paire de clés privée /publique de la CA OpenSSH

ssh-keygen permet de générer un paire de clés privée / publique pour gérer la CA OpenSSH. Les clés seront enregistrées dans le dossier **/etc/ssh** accessible uniquement avec les **droits root**.

Création en précisant :

- (facultatif) le **type** de clé (rsa, dsa, ecdsa ou ed25519) avec le paramètre **-t** (par défaut le type est rsa) ;
- (facultatif) la **longueur** de la clé avec le paramètre **-b**. Par défaut la longueur est de 2048 bits.
- le nom des clés et le dossier où elles seront enregistrées avce le paramètre **-f** ;
- un commentaire avec le paramètre **-C** ;
- la passphrase avec le paramètre **-N** :

Création de la paire de clé de type ecdsa pour la CA OpenSSH dans le dossier **/etc/ssh** :

```
ssh-keygen -f /etc/ssh/ssh_ca -C "CA for SSH" -N "Sio1234*"
```

- La **clé privée** est enregistrée sous le nom **ssh_ca** ;
- La **clé publique** est enregistrée sous le nom **ssh_ca.pub** ;

- Une passphrase permet de protéger la clé privée.

Par défaut les clé sont enregistrées dans le dossier **.ssh** :

- la **clé privée** s'appelle selon le type **id_rsa**, **id_dsa**, **id_ecdsa** (c'est le cas pour la commande utilisée), **id_ed25519** ;
- la **clé publique** s'appelle selon le type **id_rsa.pub**, **id_dsa.pub**, **id_ecdsa.pub** (c'est le cas pour la commande utilisée), **id_ed25519.pub**.

Une 2e sécurité est nécessaire pour la clé privée : il est nécessaire de **changer les droits d'accès** pour que seul le compte puisse l'utiliser :

```
chmod 600 .ssh/id_rsa
```

Format d’une clé publique ssh

Une clé publique SSH est une chaîne de texte en une seule ligne, avec plusieurs parties distinctes : <type> <clé encodée en base64> <commentaire facultatif>

Type : Spécifie l'algorithme utilisé pour générer la clé. Par exemple :

- ssh-rsa : clé utilisant l'algorithme RSA.
- ecdsa-sha2-nistp256 : clé ECDSA avec une courbe elliptique.
- ssh-ed25519 : clé basée sur l'algorithme Ed25519.

Clé encodée en base64 : chaîne encodée en Base64 représentant la clé publique.

Commentaire facultatif : information pour identifier la clé : nom d'utilisateur et l'hôte depuis lequel la clé a été générée.

Génération d'un certificat pour un serveur

La clé privée de la CA (/etc/ssh/ssh_ca) va signer la clé publique du serveur (clé générée automatiquement lors de l'installation du service SSH). Pour cet exemple c'est la clé ecdsa qui sera signée.
Utilisation des paramètres suivants :

- paramètre **-s** pour indiquer la clé privée de la CA qui va **signer** le certificat du serveur :
- paramètre **-I** pour indiquer l'**identifiant du certificat** (Key ID).
  * paramètre -h pour indiquer un certificat serveur
- paramètre -z pour indiquer u numéro de série
- paramètre -n pour indiquer un ou plusieurs nom/@ip de machines

```
$ ssh-keygen -s /etc/ssh/ssh\_ca -I SSH-SERVEUR1-CERT \
```

```
-h -z 1234 \
-n localhost,10.0.0.102,www.serveurweb.fr \
/etc/ssh/ssh\_host\_ecdsa\_key.pub
```

Le certificat Serveur **/etc/ssh/ssh_host_ecdsa_key-cert.pub** a été créé.

Le paramètre -L permet d'afficher le contenu du certificat :

```
ssh-keygen -L -f /etc/ssh/ssh\_host\_ecdsa\_key-cert.pub
/etc/ssh/ssh_host_ecdsa_key-cert.pub:
        Type: ecdsa-sha2-nistp256-cert-v01@openssh.com host certificate
        Public key: ECDSA-CERT SHA256:gncg0I1AzJaivFQL5mA7V2H/5c8CSY4kZEdyuCK6ySQ
        Signing CA: ECDSA SHA256:t225t0te+LcQXCo1og6TYAnD2PIQ/vnMo/nWL64bIaY (using ecdsa-sha2-nistp256)
        Key ID: "SSH-SERVEUR1-CERT"
        Serial: 1234
        Valid: forever
        Principals:
                localhost
                10.0.0.102
                www.serveurweb.fr
        Critical Options: (none)
        Extensions: (none)
```

Commentaires :

- le champ **Type** indique un certificat serveur **host** ;
- l'identifiant **Key ID** est **SSH-SERVEUR1-CERT** ;
- il n'y a pas d'extensions autorisant telle ou telle fonctionnalité car ne s'applique qu'à un certificat utilisateur
- l'empreinte de la clé publique **Public key** est différente de celle qui a signée le certificat **Signing CA**
- Il n'y a pas de date de validité.

Le serveur doit être configuré pour présenter ce certificat à tout client qui voudra se connecter.

Pour cela modifier le fichier du service SSH **/etc/ssh/sshd_config** pour

- ajouter la directive **HostCertificate** :
- indiquer également d'utiliser la clé privée **ssh_host*ecdsa*key** avec la directive **HostKey** :

```
# utiliser un certificat serveur
HostKey /etc/ssh/ssh_host_ecdsa_key
HostCertificate /etc/ssh/ssh_host_rsa_key-cert.pub
```

Le lancement en mode debug du service ssh montre le chargement de la clé privée et du certificat :

```
root@Serveur1:~# /usr/sbin/sshd -d
...
debug1: private host key #0: ecdsa-sha2-nistp256 SHA256:gncg0I1AzJaivFQL5mA7V2H/5c8CSY4kZEdyuCK6ySQ
debug1: host certificate: #0 type 6 ECDSA-CERT
...
```

Redémarre le service SSH sur le serveur

```
systemctl restart ssh
```

Il est maintenant possible pour un client SSH de se connecter à un serveur présentant un certificat signé par la CA :

- sans devoir valider l'identité du serveur ;
- sans enregistrement d'une nouvelle ligne dans le fichier **known_hosts**.

Pour indiquer qu'un serveur particulier, présentant un certificat sign par la CA OpenSSH n'est plus de confiance, il suffit d'ajouter dans le fichier **known_hosts** une ligne avec le marqueur **@revoked** associé à la clé publique du serveur.

Mise en place du certificat serveur au niveau du client

La configuration du client permet à celui-ci de faire confiance à n'importe quel serveur qui présente un certificat signé par une CA préalablement définie.

Pour cela, le fichier **~/.ssh/known_hosts** doit contenir une ligne avec la directive **@cert-authority** et la clé publique de la CA qui est reconnue.

Le format d'une ligne est le suivant avec des quatre types d'information séparés par des espaces :

- markers (optional),
- hostnames (une liste de valeurs séparées par des virgules, avec utilisation possibles des caractères jocker * et ? ;
- public key (clé publique de la CA),
- comment (commentaires)

Récupérez la clé publique de la CA et ajoutez-là dans le fichier **know_hosts** puis compétez la ligne ajoutée pour ajouter les paramètres nécessaires (directive **@cert-authority**, hostnames). Cela devfraot resmebler à cela :

```
@cert-authority 10.* ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBPsbCPdkmh9SXN2b/WwU9dqQi7z+0W5YY3SblMHdIS7gkQ2/YcFf
Cs/f56cwIDc2pkNmuVoFDJPo2gg5khqYtDg= CA for SSH
```

En se connectant en mode verbeux, vous visualisez que le certifivatt présenté a bien été validé :

```
debug1: Server host certificate: ecdsa-sha2-nistp256-cert-v01@openssh.com
SHA256:gncg0I1AzJaivFQL5mA7V2H/5c8CSY4kZEdyuCK6ySQ, serial 1234 ID "SSH-SERVEUR1-CERT" CA ecdsa-sha2-
nistp256 SHA256:t225t0te+LcQXCo1og6TYAnD2PIQ/vnMo/nWL64bIaY valid forever
debug1: load_hostkeys: fopen /home/charles/.ssh/known_hosts2: No such file or directory
debug1: load_hostkeys: fopen /etc/ssh/ssh_known_hosts: No such file or directory
debug1: load_hostkeys: fopen /etc/ssh/ssh_known_hosts2: No such file or directory
debug1: Host '10.0.0.102' is known and matches the ECDSA-CERT host certificate.
debug1: Found CA key in /home/charles/.ssh/known_hosts:1
```

Génération d'un certificat utilisateur autosigné

```
$ ssh-keygen -s id_rsa -I charles-CERT id
```

Le paramètre -L permet d'afficher le contenu du certificat :

```
ssh-keygen -L -f id_rsa-cert.pub
id_rsa-cert.pub:
        Type: ssh-rsa-cert-v01@openssh.com user certificate
        Public key: RSA-CERT SHA256:Dv79U+oPIbv5CGIiROMxM02479DZwaj3xOcYJKw1Bww
        Signing CA: RSA SHA256:Dv79U+oPIbv5CGIiROMxM02479DZwaj3xOcYJKw1Bww (using rsa-sha2-512)
        Key ID: "charles"
        Serial: 0
        Valid: forever
        Principals: (none)
        Critical Options: (none)
        Extensions:
                permit-X11-forwarding
                permit-agent-forwarding
                permit-port-forwarding
                permit-pty
                permit-user-rc
```

Commentaires :

- le champ **Type** indique un certificat utilisateur **user**
- l'identifiant **Key ID** est charles
- l'empreinte de la clé publique **Public key** est la même que celle qui a signée le certificat **Signing CA**

Génération d'un certificat utilisateur signé par la CA OpenSSH

Génération du certificat utilisateur

Les certificats OpenSSH permettent d'ajouter :

- des informations sur l'**identité** du propriétaire de la clé publique
- et des **contraintes de validité**.

Pour signer un certificat avec ssh-keygen, deux paramètres au minimum sont nécessaires :

- paramètre **-s** pour indiquer la clé privée qui va **signer** : sa propre clé privée ou celle de la CA OpenSSH
- le paramètre **-I** pour indiquer l'**identifiant du certificat** (Key ID).

Il est cependant indispensable de préciser :

- une ou plusieurs valeurs de **principals** avec le paramètre -n chaque **principal** est un nom d'utilisateur dotn on peut prendre l'identité

Le nom du fichier de certificat est généré en ajoutant **-cert.pub** au nom de la clé publique. Sur le serveur ayant les clé de la CA OpenSSH, signez la clé publique de l'utilisateur :

- transférez la clé publique de l'utilisateur sur le serveur
- signez la clé publique avec la clé privée de la CA **/etc/ssh/ssh_c**.

```
ssh-keygen -s /etc/ssh/ssh_ca -I CHARLES-CERT \
-z 1
-n charles
 id_rsa.pub
```

- transférez le certificat sur le poste du client dans son dossier **.ssh**

Le paramètre -L permet d'afficher le contenu du certificat :

```
ssh-keygen -L -f /home/charles/id_ecdsa-cert.pub
/home/charles/id_ecdsa-cert.pub:
        Type: ecdsa-sha2-nistp256-cert-v01@openssh.com user certificate
        Public key: ECDSA-CERT SHA256:vq5uCGtqvFGK2CiBX07gtXvfZuGklvnrQ2IBqEoiuzA
        Signing CA: ECDSA SHA256:t225t0te+LcQXCo1og6TYAnD2PIQ/vnMo/nWL64bIaY (using ecdsa-sha2-nistp256)
        Key ID: "CHARLES-CERT"
        Serial: 1
        Valid: forever
        Principals:
                charles
        Critical Options: (none)
        Extensions:
                permit-X11-forwarding
                permit-agent-forwarding
                permit-port-forwarding
                permit-pty
                permit-user-rc
```

Commentaires :

- le champ **Pricipals** indique le nom d'utilisateur dont qui sera présenté au serveur
- l'identifiant **Key ID** est CHARLES-CERT
- l'empreinte de la clé publique **Public key** est différente de celle qui a signée le certificat **Signing CA**

Validation d'un certificat de client SSH

Il y a plusieurs méthodes :

- **méthode globale** qui s'applique à tous les utilisateurs ;
- **méthode personnalisée** spécifique à un client ;
- une **combinaison** de ces deux méthodes.

Validation généralisée de certificats

- méthode à privilégier quand la majorité d'utilisateurs utilisent des certificats OpenSSH signés par une CA connue de l'administrateur.

- indication sur le serveur de la liste de ces clés de CA de confiance,
- les utilisateurs
    - n'ont plus besoin de maintenir un fichier **.ssh/authorized_keys** ;
    - utilisent des certificats contenant un ou plusieurs **principals** ;
    - accèdent aux comptes utilisateurs présents dans la liste des **principals** de leur certificat.

Pour activer la validation généralisée des certificats, le fichier de configuration du serveur SSH **/etc/ssh/sshd_config** doit contenir la directive **TrustedUserCAKeys** qui précise le nom du fichier contenant la liste des clef publique des CA OpenSSH.

```
TrustedUserCAKeys /etc/ssh/ssh_ca_keys
```

> **IMPORTANT** : le certificat du client ne sera considéré que si sa liste de nom-clés (principal) contient le nom du compte auquel il tente de se connecter.
>
> SAUF si un fichier spécifique à chaque compte indique la liste des noms-clés acceptés. Directive **AuthorizedPrincipalsFile** dans **sshd_config**

- Redémarrez ensuite le service SSH

> L'utilisateur peut maintenant se connecter en SSH au serveur sans avoir au préalable fait renseigner sa clé publique dans le fichier **authorized_keys** du serveur

Mise en place côté client

Quand un client accepte pour la première fois la connexion à un serveur, il enregistre dans le fichier **~/.ssh/know_hosts** une ligne avec la clé publique du serveur.

Lors d'une prochaine connexion, la clé présentée par le serveur sera **comparée** à celle déjà enregistrée afin d'éviter les attaques du type **homme du milieu**.

Le client gère la liste des serveurs connus dans son fichier **~/.ssh/hnow_hosts** :

- il y a une ligne par serveur sur lequel le client a accepté de se connecter ;
- chaque ligne contient l'adresse IP ou le nom DNS du serveur et la clé publique du serveur ;
- le fichier peut être haché pour masquer l'adresse IP/nom DNS ;
- la commande suivante permet de rechercher dans la liste une clé particulière

```
ssh-keygen -F @IP
```

- le paramètre **-R** permet d'effacer une clé et le paramètre **-r** permet d'afficher l'empreinte SSHFP (SSH FingerPrint).

Le client doit faire signer sa clé publique pour obtenir un certificat qui doit être placé dans le même répertoire que sa clé privé et publique.

```
cp client_key ~/.ssh/
cp client_cert.pub ~/.ssh/
```

Les permissions doivent être correctes :

```
chmod 600 ~/.ssh/client_key
chmod 644 ~/.ssh/client_cert.pub
```

Lors d’une connexion avec la clé privée, le certificat sera automatiquement présenté au serveur.

Le client ssh doit seulement indique quelle clé privée utiliser avec l’option -i

```
ssh -i ~/.ssh/client_key nomDNSserveur
```

Ainsi même s’il n’y a pas de clé publique client dans authorized_keys, l’authentification se fait. L’option -v permet de voir l’utilisation du certificat

```
ssh -v -i ~/.ssh/client_key nomDNSserveur
```

Vous pouvez utiliser aussi le fichier de configuration du client ssh

```
Host <nom_du_serveur>
    HostName <adresse_du_serveur>
    User <nom_utilisateur>
    IdentityFile ~/.ssh/client_key
    CertificateFile ~/.ssh/client_cert.pub
```

Les commandes utiles

- obtenir la clé publique à partir de la clé privée d'une identité utilisateur au format openSSH

```
$ ssh-keygen -y -f utilisateur-identite.pem > id_rsa.pub
```

- Obtenir la clé publique à partir du certificat de l'utilisateur au format openSSH pour une connexion ssh
    - Extraire la clé publique du certificat de l'utilisteur

```
$ openssl x509 -in charles-cert.pem -pubkey –noout > id_rsa.pem
```

noout permet d'avoir uniquement la clé publique sans le certificat

- Convertir la clé publique du format PEM au format openssh

```
$ ssh-keygen -i -m PKCS8 -f id_rsa.pem > id_rsa.pub
```

- Obtenir la clé publique de la CA du certificat de la CA au format openSSH pour une connexion ssh
    - Extraire la clé publique du certificat du CA

```
$ openssl x509 -in pkicub-cert.pem -pubkey –noout > ca.pem
```

```
* Convertir la clé publique du format PEM au format openssh
```

```
$ ssh-keygen -f ca.pem > ca.pub
```

- Vérifiez les journaux SSH pour voir si le certificat est utilisé correctement :

```
journalctl -u ssh
```

- Sur le clien, ajoutez l'option -v pour obtenir plus de détails sur le processus de connexion :

```
ssh -v <nom_du_serveur>
```

- Facilité d'utilisation avec plusieurs serveurs en faisant confiance à la même CA en ajoutant la clé publique de la CA à ~/.ssh/known_hosts en tant qu'autorité de certification :

```
echo "@cert-authority *.example.com ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQE..." >> ~/.ssh/known_hosts
```

Cela permet au client de faire confiance à tous les serveurs avec des certificats signés par cette CA.

- Visualiser le contenu d'un certificat

```
openssl x509 -in filename.pem -text –noout
```

-noout pour ne pas afficher en base64 qui permet d'encode des données binaires ASCII

- emetteur du certificat : -issuer

```
openssl x509 -in filename.pem -issuer –noout
```

- Sujet du certificat : -subject

```
openssl x509 -in filename.pem -subject –noout
```

- Empreinte digitale : -fingeprint

```
openssl x509 -in filename.pem -fingerprint –noout
```

- Clé publique : -pubkey

```
openssl x509 -in filename.pem –pubkey -noout –out id_rsa.pub
```

Notes

Configurer une authentification SSH avec des certificats (et non simplement des clés publiques/privées) est une méthode avancée qui repose sur un serveur d'autorité de certification (CA) pour signer les clés SSH. Voici un guide étape par étape pour mettre cela en place :

1. Générer une paire de clés pour l'utilisateur Sur la machine client (utilisateur) :

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa_cert
```

Cela crée :

- id*rsa*cert (clé privée)
- id*rsa*cert.pub (clé publique)

2. Créer une autorité de certification (CA) Sur la machine administrateur (ou serveur CA) :

```
ssh-keygen -f /etc/ssh/ssh_ca -C "CA for SSH" -N ""
```

Cela crée :

- /etc/ssh/ssh*ca (clé privée de la CA)* * */etc/ssh/ssh*ca.pub (clé publique de la CA)

3. Signer la clé publique de l'utilisateur Toujours sur la machine CA :

```
ssh-keygen -s /etc/ssh/ssh_ca -I user_cert -n nom_utilisateur -V +52w ~/.ssh/id_rsa_cert.pub
```

- -I : identifiant du certificat
- -n : nom d'utilisateur autorisé
- -V : durée de validité (ici 52 semaines)

Cela génère un fichier id*rsa*cert-cert.pub.

Visualiser le contenu du certificat :

```
ssh-keygen -L -f id_rsa_cert.pub
```

1. Configurer le serveur SSH pour accepter les certificats Sur le serveur SSH (où l'utilisateur veut se connecter), ajoutez dans /etc/ssh/sshd_config :

```
TrustedUserCAKeys /etc/ssh/ssh_ca.pub
```

Puis redémarrez le service SSH :

```
sudo systemctl restart sshd
```

1. Configurer le client SSH Sur la machine client, dans ~/.ssh/config :

```
Host monserveur
    HostName monserveur.exemple.com
    User nom_utilisateur
    IdentityFile ~/.ssh/id_rsa_cert
    CertificateFile ~/.ssh/id_rsa_cert-cert.pub
```

1. Connexion

```
ssh monserveur
```

Script Bash automatisé pour configurer une authentification SSH avec des certificats :

Ce que fait le script :

- Génère une paire de clés RSA pour l'utilisateur.
- Crée une autorité de certification (CA) SSH.
- Signe la clé publique de l'utilisateur avec la CA.
- Configure le serveur SSH pour faire confiance à la CA.
- Crée un fichier ~/.ssh/config pour simplifier la connexion.

Utilisation :

- Télécharge le script.
- Rendre exécutable :

```
chmod +x configurer_authentification_ssh.sh
```

- contenu du script : <code> #!/bin/bash

  Script d'automatisation de l'authentification SSH avec certificat

  s # À exécuter avec les droits root ou sudo

=== Étape 1 : Générer une paire de clés pour l'utilisateur ===

echo “[1/5] Génération de la paire de clés utilisateur…” USER*KEY*DIR=“$HOME/.ssh” USER*KEY*NAME=“id*rsa*cert” mkdir -p “$USER*KEY*DIR” ssh-keygen -t rsa -b 4096 -f “$USER*KEY*DIR/$USER*KEY*NAME” -N “”

=== Étape 2 : Créer une autorité de certification (CA) ===

echo “[2/5] Création de l'autorité de certification…” CA*DIR=“/etc/ssh” CA*KEY*NAME=“ssh*ca” sudo ssh-keygen -f “$CA*DIR/$CA*KEY_NAME” -C “CA for SSH” -N “”

=== Étape 3 : Signer la clé publique de l'utilisateur ===

echo “[3/5] Signature de la clé publique utilisateur…” CERT*ID=“user*cert” USERNAME=“$(whoami)” VALIDITY=“+52w” sudo ssh-keygen -s “$CA*DIR/$CA*KEY*NAME” -I “$CERT*ID” -n “$USERNAME” -V “$VALIDITY” “$USER*KEY*DIR/$USER*KEY*NAME.pub”

=== Étape 4 : Configurer le serveur SSH pour accepter les certificats ===

echo “[4/5] Configuration du serveur SSH…” sudo bash -c “echo 'TrustedUserCAKeys $CA*DIR/$CA*KEY*NAME.pub' » /etc/ssh/sshd*config” sudo systemctl restart sshd

=== Étape 5 : Créer un fichier de configuration SSH côté client ===

echo “[5/5] Création du fichier de configuration SSH client…” CONFIG*FILE=“$USER*KEY*DIR/config” SERVER*ALIAS=“monserveur” SERVER_HOST=“monserveur.exemple.com”

cat «EOF

> "$CONFIG_FILE"

H

ost $SERVER*ALIAS HostName $SERVER*HOST

```
  User $USERNAME
  IdentityFile $USER_KEY_DIR/$USER_KEY_NAME
  CertificateFile $USER_KEY_DIR/${USER_KEY_NAME}-cert.pub
```

EOF

chmod 600 “$CONFIG_FILE”

echo “ Configuration terminée. Vous pouvez maintenant vous connecter avec : ssh $SERVER_ALIAS”

</code>