sshd

Service ssh

OPTIONS

-4 Force l'utilisation d'adresses IPv4 uniquement

-6 Force l'utilisation d'adresses IPv6 uniquement

-C con_spec Spécifie les paramètres de connexion à utiliser pour le mode test étendu -T. Si fournis, toute directive Match dans la configuration qui s'applique à l'utilisateur, hôte, et adresse spécifié sont définis avant d'écrire la configuration sur stdout.

-c host_certificate_file Spécifie le chemin du certificat. Il doit matcher un fichier de clé hôte spécifié par -h ou HostKey

-d Ne lance pas en tâche de fond

-d mode debug

-E log_file Ajoute les logs de debuggage à ce fichier au lieu des logs système

-e Écrit les logs de debuggage sur stderr au lieu des logs système

-f config_file Spécifie le nom d'un fichier de configuration. Défaut : /etc/ssh/sshd_config.

-g login_grace_time Délai de grâce pour que les clients puissent s'authentifier. Défaut : 120 secondes. 0 = aucune limite.

-h host_key_file Fichier depuis lequel la clé hôte est lue. Doit être donné si sshd ne tourne pas en root. Défaut : /etc/ssh/ssh_host_dsa_key, /etc/ssh/ssh_host_ecdsa_key, /etc/ssh/ssh_host_ed25519_key et /etc/ssh/ssh_host_rsa_key.

-i Spécifie que sshd est lancé depuis inetd

-o option Peut être utilisé pour donner des options au format utilisé dans le fichier de configuration

-p port Port d'écoute. Défaut : 22. Plusieurs ports sont permis. Si spécifié, la directive Port est ignorée. Les ports spécifié en utilisant ListenAddress remplace les ports de la ligne de commande

-q Mode silencieux. N'envoie rien aux logs système

-T Mode test étendu. Vérifie la validité du fichier de configuration, affiche la configuration effective sur stdout puis quitte.

-t Mode test. Vérifie la validité du fichier de configuration et les clés.

-u len Spécifie la taile du champs dans la structure utmp qui maintient le nom d'hôte distant. Si le nom d'hôte résolu est supérieur à len, la valeur décimale est utilisé à la place. Cela permet aux hôtes avec un nom d'hôte très long d'être identifiés de manière unique.

Authentification

sshd support le protocole ssh v2 uniquement. Chaque hôte a une clé spécifique, utilisée pour identifier l'hôte. Quand un client se connecte, le service répond avec sa clé hôte publique. Le client compare la clé avec sa base pour vérifier qu'elle n'a pas changé. Un agrément de clé Diffie-Hellman est utilisé pour générer une clé de session partagée. Le reste de la session est chiffrée en utilisant un chiffrement symétrique, actuellement AES-128, Blowfish, 2DES, CAST128, Arcfour, AES-192, ou AES-256. Le client sélectionne l'alogrithme à utiliser parmis ceux offerts par le serveur. Additionnellement, l'intégrité de session est fournie via un code MAC (hmac-md5, hmac-sha1, umac-64, umac-128, hmac-sha2-256 ou hmac-sha2-512).

Finalement, le serveur et le client entrent dans un dialogue d'authentification. Le client tente de s'authentifier lui-même en utilisant une authentification basé sur l'hôte, authentification à clé publique, authentification par challenge-réponse, ou authentification par mot de passe. Si le client réussi à s'authentifier, il entre en préparation d'ouverture de session. À ce moment le client peut demander des éléments tel qu'un pseudo-tty, des connexions de forwarding X11, connexions forwarding TCP, ou forwarder la connexion de l'agent d'authentification via le canal sécurisé.

Après cela, le client demande soit un shell, ou l'exécution d'une commande. Les partis entrent en mode session. Dans ce mode, un des partis peut envoyer des données, donc les données sont envoyer depuis/vers le shell ou la conmmande dans le serveur, et le terminal côté client. Quand le programme utilisateur se termine et que toutes les connexions sont fermées, le serveur envoie un status de sortie au client, et les 2 parties se terminent.

Processus de connexion

Quand un utilisateur se connecte, sshd effectue les étapes suivantes :

1. si le login est dans un tth, et qu'aucune commande n'a été spécifiée, affiche la date de dernière connection et /etc/motd
2. Si le login est dans un tth, enregistre l'horodatage de connexion
3. Véfirie /etc/nologin ; s'il existe, affiche son contenu et quitte (sauf root)
4. Change pour s'exécuter avec des privilèges utilisateurs normaux
5. Définis un environnement basique
6. Lit ~/.ssh/environment, s'il existe, et les utilisateurs sont autorisés à changer leur environnement.
7. Change le répertoire personnel de l'utilisateur
8. Si ~/.ssh/rc existe, et que l'option PermitUserRC est définis, le lance, sinon si /etc/ssh/sshrc exist, le lance, sinon lance xauth.
9. Lance le shell ou la commande de d'utilisateur. Toutes les commandes sont lancées sous le shell de login de l'utilisateur.

sshrc

Si le fichier ~/.ssh/rc existe, sh le lance après avoir lu les fichiers d'environnement, mais avant de lancer la commande ou le shell de l'utilisateur. Il ne doit pas produire de sortie sur stdout ; stderr doit être utilisé à la place. Si le forwarding X11 est utilisé, il reçoit la paire "proto cookie" dans son entrée standard. Le script doit appeler xauth parce que sshd ne lance pas xauth automatiquement pour ajouter les cookies X11.

Le but premier de ce fichier est de lancer des routines d'initialisation qui peuvent être nécessaires avant que le répertoire personnel de l'utilisateur devienne accessible. AFS est un exemple particulier d'un tel environnement.

Ce fichier contient probablement du code d'initialisation suivi par quelquechose de similaire à :

```
if read proto cookie && [ -n "$DISPLAY" ]; then
 if [ `echo $DISPLAY | cut -c1-10` = 'localhost:' ]; then
  # X11UseLocalhost=yes
  echo add unix:`echo $DISPLAY | cut -c11-` $proto $cookie
 else
  # X11UseLocalhost=no
  echo add $DISPLAY $proto $cookie
 fi | xauth -q -
fi
```

Si ce fichier n'existe pas, /etc/ssh/sshrc est lancé, et s'il n'existe pas, xauth est utilisé pour ajouter le cookie.

Fichier authorized_keys

AuthorizedKeysFile spécifie les fichiers contenant les clé publique pour l'authentification. Si cette option n'est pas spécifiée, le défaut est ~/.ssh/autorized_keys et ~/.ssh/authorized_keys2. Chaque ligne du fichier contient une clé. Les clé publiques consistent des champs

suivants : options, keytype, base64-encoded key, comment. Le champ option est optionnel. Le keytype est “ecdsa-sha2-nistp256”, “ecdsa-sha2-nistp384”, “ecdsa-sha2-nistp521”, “ssh-ed25519”, “ssh-dss” ou “ssh-rsa”.

Noter que chaque ligne peut faire des centaines d’octets de longs, jusqu’à une limite de 8Ko, ce qui permet des clé DSA jusqu’à 8Ko, et des clé RSA jusqu’à 16Ko.

sshd force une clé RSA minimal de 768 bits. Les options, si présentes, sont séparés par ’,’. Aucun espace n’est permis, excepté entre guillemets double. Les options suivantes sont supportés :

agent-forwarding Autorise le forwarding de l’agent d’authentification précédemment désactivé par l’option restrict

cert-autority Spécifie que la clé est une autorité de certification qui sert à valider les certificats utilisateurs

command="command" Cmmande exécutée quand cette clé est utilisée. Peut être utile pour restreindre une clé a effectuer une opération spécifique.

environment="NAME=value" Définie une variable d’environnement

from="pattern-list" Spécifie que le nom canonique de l’hôte distant ou son IP doivent être présents dans cette liste.

no-agent-forwarding Interdit le forwarding de l’agent d’authentification

no-port-forwarding Interdit le forwarding TCP

no-pty Empêche l’allocation tty

no-user-rc Désactive l’exécution de ~/.ssh/rc

no-X11-forwarding Interdit le forwarding X11

permitopen="host :port" Limite le forwarding de port de ssh -L, à l’hôte :port spécifié.

port-forwarding Active le forwarding précédemment désactivé par l’option restrict

principals="principals" Liste de principaux pour l’authentification

pty Autorise l’allocation tty précédemment désactivé par l’option restrict

restrict Active toutes les restrictions, forwarding, pty, ~/.ssh/rc.

tunnel="n" Force un périphérique tun dans le serveur

user-rc Autorise l’exécution de ~/.ssh/rc précédemment désactivé par l’option restrict

X11-forwarding Autorise le forwarding X11 précédemment désactivé par l’option restrict

Fichier ssh_known_hosts

Les fichier /etc/ssh/ssh_known_hosts et ~/.ssh/known_hosts contiennent les clés publiques pour tous les hôtes connus. Le fichier global devrait être préparé par l’administrateur, et le fichier par utilisateur est maintenu automatiquement. Chaque ligne dans ces fichier contient les champs suivants : marquer, nom d’hôte, type de clé, clé encodé en base64, commentaire. Les champs sont séparés par des espaces.

Le marqueur est optionnel, mais s’il est présent il doit être sous la forme ’@cert-authority’ pour indiquer que la ligne contient un certificat d’autorité, ou ’@revoked’ pour indiquer que la clé contenu dans la ligne est révoquée.

Les noms d’hôte sont une liste de motifs séparé par ’,’ ; chaque pattern est matché avec le nom canonique de l’hôte (en authentifiant un client) ou avec le nom du l’utilisateur (en authentifiant un serveur). A pattern peut inclure ’*’, ’ ?’, et ’ !’ pour la négation.

Alternativement, les noms d’hôte peuvent être stockés de manière hashé pour cacher les noms d’hôte ou les adresse.

Le type de clé et la clé encodé en base64 sont pris directement depuis la clé hôte ; ils peuvent être obtenus, par exemple, depuis /etc/ssh/ssh_host_rsa_key.pub.

Il est permis, mais non recommandé, d’avoir plusieurs lignes ou différentes clés hôte pour le même nom. Cela se produit quand des formes courte de noms d’hôte pour différents domaines sont placés dans le fichier. Il est possible que les fichiers contenant des informations en conflit acceptent l’authentification si l’information valide peut être trouvée dans le fichier.

Exemple de fichier ssh_known_hosts :

```
# Comments allowed at start of line
closenet,...,192.0.2.53 1024 37 159...93 closenet.example.net
cvs.example.net,192.0.2.10 ssh-rsa AAAA1234.....=
# A hashed hostname
|1|JfKTdBh7rNbXkVAQCRp4OQoPfmI=|USECr3SWf1JUPsms5AqfD5QfxkM= ssh-rsa
AAAA1234.....=
# A revoked key
@revoked addentry articles autoadd autofind autoprod createalpha createbeta createdb createprod findentry
fullpowa generator.php genhtml genman genmd gentex html images insert man md pdf regen setfor setfor2 sql
temp tex threads ToDo ssh-rsa AAAAB5W...
# A CA key, accepted for any host in *.mydomain.com or *.mydomain.org
@cert-authority *.mydomain.org,*.mydomain.com ssh-rsa AAAAB5W...
```

Fichiers

~/hushlogin Supprime l'affichage de la date de dernière connexion et /etc/motd

~/.rhosts Utilisé pour l'authentification basé sur l'hôte

~/.shosts idem .rhosts, mais sans login permit avec rlogin/rsh

~/.ssh/ Répertoire de configuration par défaut pour le utilisateurs

~/.ssh/authorized_keys Liste des clés publiques qui peuvent être utilisé pour se connecter avec cet utilisateur.

~/.ssh/environment Ce fichier est lu dans l'environnement au login

~/.ssh/known_hosts Contient une liste de clés hôte pour tous les hôtes sur lesquels l'utilisateur s'est connecté

~/.ssh/rc Exécuté par ssh quand l'utilisateur se logs, juste avant de lancer le shell ou la commande

/etc/hosts.equiv Fichier pour l'authentification basé sur l'hôte. doit être writable par root uniquement

/etc/shosts.equiv idem sans autoriser rlogin/rsh

/etc/moduli Contient les groupes Diffie-Hellman utilisé pour l'échange de clé DH.

/etc/motd Message Of The Day

/etc/nologin Si ce fichier existe, sshd refuse la connexion excepté pour root

/etc/ssh/ssh_host_dsa_key

/etc/ssh/ssh_host_ecdsa_key

/etc/ssh/ssh_host_ed25519_key

/etc/ssh/ssh_host_rsa_key Contient la clé privée de l'hôte

/etc/ssh/ssh_host_dsa_key.pub

/etc/ssh/ssh_host_ecdsa_key.pub

/etc/ssh/ssh_host_ed25519_key.pub

/etc/ssh/ssh_host_rsa_key.pub Contient la partie publique de la clé de l'hôte

/etc/ssh/ssh_known_hosts Fichier système listant les clé hôte connues

/etc/ssh/sshd_config Fichier de configuration pour sshd

/etc/ssh/sshrc Similaire à ~/.ssh/rc

/var/empty Répertoire chroot utilisé par sshd durant la séparation de privilège dans la phase de préauthentification. Ce répertoire ne devrait pas contenir de fichiers et doit être possédé par root.

/var/run/sshd.pid fichier pid du processus sshd.