ssh

programme de connexion distante

OPTIONS

-4 Force l'utilisation d'IPv4 uniquement

-6 Force l'utilisation d'IPv6 uniquement

-A Autorise le forwarding de l'agent d'authentification. Peut être spécifié par hôte dans un fichier de configuration.

-a Désactive le forwarding de l'agent d'authentification

-b bind_address Utilise l'adresse locale spécifiée pour la connexion

-C Demande la compression des données. Utile pour les connexions lentes, mais ralentit le traffic dans les réseaux rapide

-c cipher_spec Liste de chiffrements dans l'ordre de préférence pour le chiffrement de la session

-D [bind_address :]port Spécifie un port forwarding locale dynamique. Cela fonctionne en allouant un socker d'écoute sur le port côté local, optionnellement lié à l'adresse spécifiée. Quand une conexion est faite sur ce port, la connexion est envoyés au canal sécurisé, et le protocole d'application est ainsi utilisé pour déterminer où se connection depuis la machine distante. Actuellement les protocoles SOCKS4 et SOCKS5 sont supportés, et ssh agit comme un serveur SOCKS.

-E log_file Ajoute les logs débug à log_file au lieu de l'erreur standard

-e escape_char Définis le caractère d'échappement pour les sessions avec un pty (défaut : ~). Ce caractère est seulement reconnu au début d'une ligne. Ce caractère suivi par un '.' ferme la connexion. Suivi par control-Z suspend la connexion, et suivi par lui-même envoie le caractère échappé une fois.

-F configfile Spécifie un fichier de configuration par utilisateur alternatif. Défaut : ~/.ssh/config

-f ssh se place en tâche de fond juste avant l'exécution

-G Affiche sa configuration après avoir évalué Host et Match, puis quitte

-g Autorise les hôte distants à se connecter aux ports forwardés. Si utilisé dans une connexion multiplexée, cette option doit être spécifiée dans le processus maître

-I pkcs11 Spécifie la librairie PKCS#11 utilisé pour communiquer avec un jeton pkcs#11

-i identity_file Fichier contenant la clé privée. Défaut : ~/.ssh/id_dsa, ~/.ssh-id_ecdsa, ~/.ssh/id_ed25519, et ~/.ssh/id_rsa.

-J [user@]host[:port] Se connecte à l'hôte spécifié en créant d'abord un saut vers cet hôte, et en établissant un forwarding TCP vers la destination. Plusieurs sauts peuvent être spécifiés séparés par des ','

-K Active l'authentification GSSAPI et le forwarding GSSAPI

-k Désactive le forwarding GSSAPI

-L [bind_address :]port :host :hostport

-L [bind_address :]port :remote_socket

-L local_socket :host :hostport

-L local_socket :remote_socket Spécifie que les connexion au port TCP ou socket unix donné dans l'hôte locla doivent être forwardés à l'hôte :port, ou socket unix donné

-l login_name Spécifie l'utilisation de connexion sur la machine distante.

-M Place le client ssh en mode master pour le partage de connexion. Peut être spécifié plusieurs fois pour ajouter une confirmation avant que les connexions esclaves soient acceptée

-m mac_spec Liste d'algorithmes MAC, dans l'ordre de préférence.

-N Ne pas exécuter une commande à distance. Utile pour un simple port forwarding

-n Redirige stdin depuis /dev/null. Doit être utilisé quand ssh tourne en tâche de fond.

-O ctl_cmd Contrôle une connexion active multiplexant un processus maître. ctl_md est interprété et passé au processus maître. (check, forwald, cancel, exit, stop)

-o option Spécifie une option tel que trouvé dans le fichier de configuration

-p port Port de connexion sur l'hôte distant

-Q query_option Affiche les algorithmes supportés pour sshv2. (cipher, cipher-auth, mac, kex, key, key-cert, key-plain, protocol-version)

-q mode silencieux

-R [bind_address :] port :host :hostport

-R [bind_address :] port :local_socket

-R remote_socket :host :hostport

-R remote_socket :local_socket Spécifie que les connexions au port TCP ou socket unix donné dans l'hôte distant doivent être forwardés à l'hôte :port ou socket unix donné, localement. Cela foncionne en alouant un socket pour écouter soit un port TCP ou un socket unix dans l'hôte distant. Quand une connexion est fait, elle est forwardée dans un canal sécurité, et une connexion est faite à l'hôte :port ou au socket, depuis la machine locale.

-S ctl_path Spécifie l'emplacement d'un socket de contrôle pour le partage de connexion, ou 'none' pour désactiver le partage de connexion

-s Peut être utilisé pour demander l'invocation d'un sous-système dans le système distant. Les sous-système facilitent l'utilisation de SSH comme transport sécurisé pour d'autres application.

-T Désactive l'allocation d'un pseudo-terminal

-t Force l'allocation d'un pseudo-terminal.

-v mode verbeux

-W host :port Demande que l'entrée et la sortie standard dans le client soient forwardés à l'hôte :port via un canal sécurisé. Implique -N, -T, ExitOnForwardFailure et ClearAllForwarding

-w local_tun [:remote_tun] Demande un forwarding avec le tunnel spécifié entre le client et le serveur

-X Active le forwarding X11.

-x Désactive le forwarding X11

-Y Active le forwarding X11 trusté, qui ne sont pas sujets aux extensions de contrôle X11 SECURITY

-y Envoie des informations en utilisant syslog, au lie de stderr

Authentification

Les méthodes disponibles pour l'authentification sont : authentification basée sur GSSAPI, authentification basé sur l'hôte, authentification à clé publique, authentification par challenge-response, et authentification par mot de passe.

L'authentification basé sur l'hôte fonctionne comme suit : si la machine sur laquelle l'utilisateur se log est listé dans /etc/hosts.equiv ou /etc/shosts.equiv dans la machine distante, et que les noms d'utilisateur sont les même des 2 côté, ou si les fichiers ~/.rhosts ou ~.shosts existent dans le répertoire personnel de l'utilisation dans la machine distante et contient une ligne contenant le nom de la machine cliente et le nom de l'utilisation dans cette machine, l'utilisateur est autorisé. Additionnellement, le serveur doit être capable de vérifie la clé hôte du client. Cette méthode d'authentification est peut sécurisé à cause de l'IP spoofing, DNS spoofing, et routing spoofing.

L'authentification à clé publique fonctionne comme suit : Le schéma est basé sur la cryptographie à clé publique, en utilisant les cryptosystems où le chiffrement et le déchiffrement sont fait en utilisant des clés séparées, et il est impossible de dériver la clé de déchiffrement depuis la clé de chiffrement. L'idée et que chaque utilisateur créé une paire de clé publique/privée pour l'authentification. Le serveur connaît la clé publique, et seul l'utilisateur connaît la clé privée. ssh supporte les algorithmes DSA, ECDSA, Ed25519 et RSA.

Le fichier ~/.ssh/authorized_keys liste les clés publiques qui sont permisses pour le logging. Quand l'utilisateur se log, ssh indique au serveur quel paire de clé utiliser pour l'authentification. Le client prouve qu'il a accès à la clé privéue et le sereur vérifie que la clé correspondante est autorisée à accéder au compte

L'utilisateur créé sa paire de clé avec ssh-keygen, stocke la clé privée dans ~/.ssh/id_dsa, /.ssh/id_ecdsa, ~/.ssh/id_ed25519, ou ~/.ssh/id_rsa. et stocke la clé publique dans ~/.ssh/id_dsa.pub, ~/.ssh/id_ecdsa.pub, ~/.ssh/id_ed25519.pub, ou ~/.ssh/id_rsa.pub.

L'utilisateur doit ensuite copier la clé publique dans ~/.ssh/authosized_keys dans le répertoire distant dans la machine distante. Ce fichier correspond au fichier rhosts conventionnel, et a une clé par ligne. Après ça, l'utilisateur peut se connecter sans donner de mot de passe

Une variation de l'authentification à clé publique est disponible sous la forme d'une authentification par certificat. Au lieu de définir des clés publique/privées, des certificats signés sont utilisé. La manière la plus simple d'utilisation cette méthode d'authentification est avec un agent d'authentification, comme ssh-agent et optionnellement a directive AddKeysToAgent.

L'authentification par challenge-réponse fonctionne comme suit : Le serveur envoie un challenge arbitraire, et demande une réponse. Des examples d'authentification par challenge-réponse incluent l'authentification BSD et PAM.

Finallement, si d'autres méthodes d'authentification échouent, ssh demande un mot de passe. Le mot de passe est envoyé à l'hôte distant pour vérification ; cependant, vu que toutes les communications sont chiffrées, le mot de passe ne peut pas être vu par quelqu'un qui écoute le réseau.

ssh maintient et vérifie automatiquement une base d'identification pour tous les hôte qu'il à vu. Les clés d'hôte sont stockés dans ~/.ssh/known_hosts. De plus, /etc/ssh/ssh_known_hosts est automatiquement vérifié pour les hôtes connus. Tout nouvel hôte est automatiquement ajouté au fichier de l'utilisateur. Si l'identification d'un hôte change, ssh alerte et désactive l'authentification par mot de passe. L'option StrictHostKeyChecking peut être utilisée pour contrôler les login sur les machine dont la clé hôte n'est pas connue ou a changé.

Quand l'identité d'un utilisateur a été acceptée par le serveur, le serveur exécute soit la commande donnée dans une session non-interactive ou, si aucune commande n'est spécifiée, donne un shell. Toute communication avec la commande distante ou le shell sont automatiquement chiffrés

Si un pseudo-terminal a été alloué l'utilisateur peut utiliser les caractèse d'échappement ci-dessous

Si aucun pseudo-terminal n'a été alloué, la session est transparent et peut être utilisée pour transférer des données binaire. Dans la plupart des systèmes, définis le caractère d'échappement à none va créer une session transparente même si un tty est utilisé

La session se termine quand la commande ou le shell se termine et que toutes les connexions TCP et X11 ont été fermées.

Caractères d'échappement

Quand un pseudo-terminal est demandé, ssh supporte certaines fonctions utilisées via un caractère d'échappement :

- **~.** Déconnexion
- **~^Z** mise en tâche de fond
- **~#** Lister les connexions forwardées
- **~&** mise en tâche de fond à la déconnexion en attendant que la connexion forwardée/session X11 se termine
- **~?** Affiche la liste des caractères d'échappement
- **~B** Envoie un BREAK au système distant
- **~C** Ouvre une ligne de commande. Actuellement, ajoute le port forwarding en utilisant les options -L, -R, et -D. Permet également d'annuel un port forwarding existant avec -KL[bind_address :]port, -KR[bind_address :]port et -KD[bind_address :]port. !command permet à l'utilisateur d'exécuter une commande locale si PermitLocalCommand est autorisée
- **~R** rekeying de la connexion
- **~V** Diminue la verbosité
- **~v** Augmente la verbosité

TCP forwarding

Le forwarding des connexions TCP via un canal sécurisé peut être spécifié soit sur la ligne de commande ou dans un fichier de configuration. Une application possible est une connexion sécurisé pour un serveur de messagerie.

Dans l'exemple ci-dessous, on cherche à chiffrer les communications entre un client IRC et un serveur, même si le serveur IRC ne supporte pas directement le chiffrement des communication. Cela foncion comme suit : l'utilisateur se connecte à l'hôte distant en utilisant ssh, en spécifiant un port à utiliser pour forwarder les connexions au serveur distant. Après ça, il est possible de démarrer le service qui est chiffré dans la machine client, en se connectant au même port local, et ssh chiffre et forward la connexion.

L'exemple suivante tunnelise une session IRC depuis la machine cliente 127.0.0.1 sur le serveur distant server.example.com :
ssh -f -L 1234 :localhost :6667 server.example.com sleep 10
irc -c '#users' -p 1234 pinky 127.0.0.1
Cela tunnélise une connexion au serveur IRC server.example.com, en joignant le canal #users, avec le pseudo pinky sur le port 1234. Le port n'est pas important du moment qu'il est supérieur à 1023, et qu'il n'est pas en conflit avec un port déjà utilisé. La connexion est redirigée au port 6667 sur le serveur distant.

L'option -f met ssh en tâche de fond et la comande distant sleep 10 est spécifiée pour permettre de démarrer le service tunnélisé.

X11 Forwarding

Si la variable ForwardX11 est à yes, et que l'utilisateur utilis X11 (la variable DISPLAY doit être définie), la connexion à l'affichage X11 est automatiquement forwardée dans le canal chiffré, et la connexion au vrai serveur X est faite depuis la machine locale.

La valeur DISPLAY définie pas ssh va pointer vers le serveur, mais avec un numéro d'affichage supérieur à 0. C'est normal parce que ssh créé un serveur X proxy sur la machine serveur pour forwarder les connexions.

ssh définis également autematiquement Xauthority sur le serveur. Il génère un cookie d'autorisation aléatoire, le stocke dans Xauthority et vérifie que toutes les connexions forwardée connaissent ce cookie et le remplace par le vrai cookie quand la connexion est ouverte. Le vrai cookie d'authentification n'est jamais envoyé au serveur.

Si la variable ForwardAgent est à yes, et que l'utilisateur utilise un agent d'authentification, la connexion à l'agent est automatiquement forwardée au serveur.

Vérifier les clés hôte

En se connectant à un serveur pour la première fois, une empreinte de la clé publique est présentée à l'utilisateur (sauf si StrictHostKeyChecking a été désactivé). Les empreintes peuvent être déterminés par ssh-keygen :
ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key
Si l'empreinte est déjà connue, elle est matchée et la clé peut être acceptée ou rejetée. Si seul les empreintes MD5 sont disponibles, ssh-keygen -E peut être utilisé pour adapter l'algorithme de l'empreinte.

Vu qu'il est difficile de comparer des clé hôte en regardant simplement les empreintes, il y a également un support pour comparer les clé hôte visuellement, en utilisant un moyen aléatoire. Avec VisualHostKey à yes, un petit graphique ASCII est affiché à chaque login au serveur. En apprenant le motif qu'un serveur connu produit, un utilisateur peut facilement voir qu'une clé a changé quand un motif complètement différent est affiché. Ces motifs pouvant être ambigûes, un motif qui semble être similaire ne donne qu'une bonne probabilité que la clé hôte soit la même, sans garantie.

Pour obtenir une liste des empreintes avec leur motif aléatoire :
ssh-keygen -lv -f ~/.ssh/known_hosts
Si l'empreinte est inconnue, une méthode alternative de vérification est disponible : les empeintes SSH vérifiés par DNS. Un enregistrement de ressource (RR), SSHFP, est ajouté à un fichier de zone et le client est capable de matche l'empreinte avec la clé présentée.

Dans cet exemple, un client se connecte au serveur, host.example.com. Le RR SSHFP doit être présent dans le fichier de zone :
ssh-keygen -r host.example.com
Les lignes affichées sont ajoutées dans la zone. Pour vérifier que la zone réponde à la demande d'empreinte :
dig -t SSHFP host.example.com
Finalement le client se connecte
ssh -o "VerifyHostKeyDNS ask" host.example.com
Voir l'option VerifyHostKeyDNS

VPN basés sur SSH

ssh support les tunnels VPN en utilisant les périphériques tun, permetant à 2 réseaux d'être joint de manière sécurisé. L'option PermitTunnel contrôle si le serveur le supporte et quel niveau.

L'exemple suivant connecte un réseaux client 10.0.50.0/24 avec un réseau distant 10.0.99.0/24 en utilisant une connection point-à-point de 10.1.1.1 vers 10.1.1.2, le serveur ssh dans la gateway vers le réseau distant 192.168.1.15, l'autorise. Dans le client :
ssh -f -w 0 :1 192.168.1.15 true
ifconfig tun0 10.1.1.1 10.1.1.2 netmask 255.255.255.252
route add 10.0.99.0/24 10.1.1.2
Dans le serveur :
ifconfig tun1 10.1.1.2 10.1.1.1 netmask 255.255.255.252
route add 10.0.50.0/24 10.1.1.1
L'accès client peut être personnalisé plus finement via /root/.ssh/authorized_keys et l'option serveur PermitRootLogin. L'entrée suivante permet les connexions dans tun1 par l'utilisateur jane, et dans tun2 par john, si PermitRootLogin est à forced-commands-only :
tunnel="1",command="sh /etc/netstart tun1" ssh-rsa ... jane
tunnel="2",command="sh /etc/netstart tun2" ssh-rsa ... john

Variables d'environnement

ssh définis les variables d'environnement suivants :

DISPLAY Emplacement du serveur X11, automatiquement définis par ssh

HOME Répertoire personnel de l'utilisateur

LOGNAME Nom de l'utilisateur

MAIL Emplacement de la boite mail de l'utilisateur

PATH PATH par défaut, tel que spécifié à la compilation de ssh

SSH_ASKPASS Si définis, exécute le programme spécifié et ouvre une fenêtre X11 pour lire la passphrase

SSH_AUTH_SOCK Identifie le chemin d'un socket UNIX utilisé pour communiquer avec l'agent

SSH_CONNECTION Identifie le client et le serveur de la connexion. Contient 4 valeurs : IP, port client, IP, port serveur.

SSH_ORIGINAL_COMMAND Contient la ligne de commande originale si une commande forcée est exécutée.

SSH_TTY Définis le nom du tty associé avec le shell ou la commande courante

TZ Affecte le timezone utilisé

USER nom de l'utilisateur

Additionnellement, ssh lit ~/.ssh/environment, voir PermitUserEnvironment

Fichiers

~/.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/config Fichier de configuration de l'utilisateur

~/.ssh/environment Contient des définitions additionnels pour les variables d'environnement

~/.ssh/id_dsa

~/.ssh/id_ecdsa

~/.ssh/id_ed25519

~/.ssh/id_rsa Clé publique pour l'authentification.

~/.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/ssh/ssh_config Fichier de configuration global.

/etc/ssh/ssh_host_key

/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_known_hosts Liste globale de clés hôtes connus. Ce fichier devrait être préparé par l'administrateur système

/etc/ssh/sshrc Exécuté par ssh quand l'utilisateur se log, juste avant de lancer le shell ou la commande.

Code de sortie

ssh quitte avec le statut de sortie de la commande distante, ou avec 255 si une erreur s'est produite.