SSH ??

SSH (Secure Shell) est un protocole permettant d'accéder à la ligne de commande d'un appareil distant sur un réseau tel qu'un routeur ou un serveur. Il permet aussi, lorsqu'activé, le transfert de ficheir entre deux hôtes avec SCP ainsi que la redirection de ports réseau pour donner ou gagner accès à des ressources distantes.

- Serveur
  - Linux
  - Windows
  - Cisco IOS
  - MikroTik RouterOS
- Client
  - Hôtes connus, chiffrage et algorithmes
  - Ligne de commande

Serveur

Pour établir une connexion entre deux systèmes, il est nécessaire de configurer un serveur auquel un client se connectera ensuite. Ceci se configure de différentes façons sur différents systèmes mais le protocole est standard et donc tous les systèmes permettant la connexion par SSH ont les mêmes éléments essentiels à configurer.

Serveur

Linux ?

Installation

Plusieurs saveurs de Linux sont distribuées avec un service SSH préinstallé qui est parfois déjà actif. D'autres distributions vous suggèreront de l'installer au moment de l'installation du système. Si le service doit être installé suite à l'installation du système, dépendamment de votre distribution, vous devrez vous référer au gestionnaire de packages ainsi qu'au répertoire logiciel de votre distribution.

Les commandes suivantes installeront et activeront SSH sur Debian et toutes les distributions découlant de Debian. Vous pouvez identifier ces distributions par le gestionnaire de packages "Aptitude" ou "apt".

```
# Installation
sudo apt update && apt install openssh-server
#
# Activation
sudo systemctl enable sshd && systemctl start sshd
```

Les commandes suivantes installeront et activeront SSH sur RHEL, Fedora ou toute autre distribution découlant de RedHat. Vous pouvez identifier ces distributions par le gestionnaire de packages "YUM" ou "DNF".

```
# Installation
sudo dnf install openssh-server
#
# Activation du service
sudo systemctl enable sshd && systemctl start sshd
```

Configuration

Le fichier de configuration du service/serveur est généralement le fichier /etc/ssh/sshd_config.

Configuration réseau

Les paramètres suivants retrouvés dans le fichier de configuration permettent de configurer les paramètres d'écoute du service sur le réseau. Le port par défaut est le port 22 et la configuration par défaut permet généralement autant aux adresses IPv4 que IPv6 de se connecter à n'importe

quelle interface réseau du serveur.

```
Port 22
AddressFamily any
ListenAddress 0.0.0.0
ListenAddress ::
```

> Quoi que le port par défaut soit 22, le changer pour un port non standard évitera généralement les attaques par force brute.

Il est possible de limiter les connexions possibles à IPv4 ou IPv6 seulement en changeant le paramètre "AddressFamily" pour inet (IPv4) ou inet6 (IPv6). Les paramètres ListenAddress indiquent à quelles adresses du serveur il est possible d'établir une connexion SSH. Lorsque le paramètre est à 0.0.0.0 ou ::, il sera possible de se connecter au serveur par toutes ses interfaces réseau. Si une adresse IP y est mentionnée, seulement cette adresse pourra être utilisée pour établir une connexion au serveur.

Configuration d'accueil et tentatives d'authentification

Il est possible de configurer les actions qui seront entreprises lors d'une connexion ou une tentative de connexion au serveur. On peut par exemple, journaliser les tentatives d'authentification ou encore afficher une bannière à l'utilisateur au moment de la connexion. Ce genre de bannière est généralement utilisé pour indiquer des conditions d'utilisations ou présenter un avertissement comme quoi l'accès à ces systèmes sont restreints.

```
Banner [chemin vers un fichier de bannière]
SyslogFacility AUTH
```

Ici, le paramètre "**Banner**" doit faire référence à un fichier contenant la bannière d'accueil à afficher aux utilisateurs tentant de s'authentifier. Ce fichier pourrait contenir un message du genre à titre d'exemple.

```
Please don't hack us. Thx.
```

Ou encore présenter un message plus menaçant indiquant aussi aux utilisateurs que les tentatives d'authentification sont journalisées.

```
+---------------------------------------------------+
| UNAUTHORIZED ACCESS TO THIS NETWORK DEVICE AND    |
| ATTACHED NETWORKS IS STRICTLY PROHIBITED.         |
| You must have explicit permission to access or    |
| configure this device. All activities performed on|
| this device may be logged or monitored without further|
```

```
| notice, and the resulting logs may be used as evidence|
| in court.                                              |
| Any unauthorized use of the system is unlawful, and    |
| may be subject to civil and/or criminal penalties.     |
+--------------------------------------------------------+
```

> Il est généralement pertinent de composer ces message en anglais pour que le plus d'acteurs tentant d'interagir avec le serveur puissent comprendre l'avertissement.

Le paramètre "**SyslogFacility**" peut être configuré avec les valeurs suivantes en fonction du niveau de sévérité des évènements qui doivent être journalisés :

- DAEMON
- USER
- AUTH
- LOCAL0 à LOCAL7 (Un niveau plus élevé inclus tous les niveaux plus bas. Ex. LOCAL7 incluera tous les messages, LOCAL1 incluera LOCAL0 et LOCAL1, etc.)
  - LOCAL0 sont pour les urgences seulement, le service est généralement non fonctionnel à ce moment
  - LOCAL1 est pour les alertes où une action doit être entreprise
  - LOCAL2 est pour les conditions critiques
  - LOCAL3 est pour les erreurs uniquement
  - LOCAL4 est pour les avertissements
  - LOCAL5 est pour les notifications
  - LOCAL6 est pour les journeaux informationels
  - LOCAL7 est pour les messages de débogage

> Il est généralement pertinent de journaliser l'ensemble des connexions, tant les connexions réussies que les connexions échouées. Ces journeaux permetteront d'identifier si le compte d'un utilisateur est compromis et exploité, si un utilisateur a accédé à un système et le nie ainsi que de détecter des tentatives d'authentification par force brute et prendre action en conséquence.

Configuration des permissions et méthodes d'authentification

Il existe deux méthodes principales d'authentification pour SSH : Authentification par clé publique (certificat) et authentification par nom d'usager et mot de passe. Par défaut, généralement tous les utilisateurs locaux du serveur ont l'autorisation de se connecter au serveur et idéalement, l'accès SSH par le compte "root" devrait être désactivée.

> Vérifiez lors du déploiment d'un nouveau serveur si le compte "root" a l'autorisation de se connecter au serveur. Ceci est généralement déconseillé car si ce compte est compromis, le

> serveur l'est aussi dans son entièreté. À moins d'un besoin spécifique, il est important de s'assurer de ceci.

L'accès par le compte "root" est géré par le paramètre "**PermitRootLogin**" qui peut avoir l'une des valeurs suivantes :

- yes

  Ce paramètre permet l'authentification tant par mot de passe que par clé publique

- no

  Ce paramètre restreint tout accès en SSH par le compte "root"

- prohibit-password (habituellement le paramètre par défaut)

  Ce paramètre permet l'authentification au compte root uniquement par une clé publique

- forced-commands-only

  Ce paramètre permet l'authentification au compte root uniquement par clé publique et restreint les commandes qu'il peut exécuter par le paramètre "**command**" qui devra aussi être spécifié dans le fichier "authorized_keys" situé dans le dossier ~/.ssh/ du compte root qui contient les clés publiques permettant l'authentification au compte en question.

Les deux autres paramètres pertinents sont "**PasswordAuthentication**" ainsi que "**PermitEmptyPasswords**" et s'expliquent d'eux même. Les valeurs possibles pour ces deux paramètres sont simplement "yes" et "no".

La gestion des autorisations pour les utilisateurs et groupes est possible avec les paramètres "**AllowUsers**", "**DenyUsers**", "**AllowGroups**" et "**DenyGroups**" suivis de soit une liste d'utilisateurs ou groupes séparés par un caractère d'espacement.

La plupart de ces paramètres peuvent être appliqués individuellements à des utilisateurs ou des groupes à l'aide du paramètre "**Match Group [liste de groupes]**" ou "**Match User [liste d'utilisateurs]**"

Configuration de services et sous-systèmes

SSH peut permettre beaucoup plus d'interactions entre un serveur et un client que d'ouvrir un shell de commande à distance. Il est possible d'établir des tunnels réseau, d'afficher une application graphique à distance (y compris un gestionnaire de fenêtres complet), de transférer des fichiers (par SCP ou par des sous-systèmes comme SFTP).

SCP est couvert plus en détail dans sa section du chapitre "clients" et permet de déposer et récupérer des fichiers à travers une session SSH. Ceci ne peut être désactivé à moins de forcer une

commande à exécuter à l'authentification SSH telle que "/bin/sh".

Les autres fonctionnalités toutefois sont paramétrables dans le fichier de configuration.

Le paramètre "AllowTcpForwarding" (ayant pour valeur "yes" ou "no") permet la redirection de ports à travers une connexion SSH. Ceci implique qu'un client peut mettre à la disposition du serveur une ressource située et accessible par le réseau du client, de mettre à sa disposition une ressource acccessible par le réseau du serveur ou même d'établir un proxy via une redirection de ports dynamique. Ceci est couvert plus en détail dans le chapitre client.

> Quoi que ceci soit dangereux car ceci permet à un attaquant de mettre à la disposition du serveur des ressources malveillantes de son côté ou de tenter d'accéder à des ressources distantes qui ne sont généralement pas accessibles à la ligne de commande (ex. page web de configuration d'un appareil), si un attaquant a accès à votre serveur et que l'exécution des commandes de l'utilisateur compromis permet d'accéder à ces ressources de toute façon, les systèmes distants seront aussi mis à la disposition d'un attaquant.

Le paramètre "**X11Forwarding**" (ayant à nouveau pour valeur "**yes**" ou "**no**") permet d'exécuter une application graphique sur un serveur distant et de l'afficher sur un "serveur" xorg du côté du client. Notez que ce paramètre fonctionne uniquement pour les applications exécutées en "X11" et ne fonctionnent généralement pas avec les applications exécutées en "Wayland".

Le(s) paramètres "**Subsystem**" permettent d'établir une connexion à différents services à travers une connexion SSH. Le sous-système sftp-server est généralement activé par défaut et nécessaire au fonctionnement de SCP2. Ce paramètre est composé d'une chaîne de caractères utilisé à la connexion SSH pour déterminer à quel exécutable la connexion sera établie.

La configuration suivante permettrait à un client d'établir une connexion à SFTP en précisant le sous-système à contacter lors de la connexion (ex. ssh -s sftp-server utilisateur@adresse.du.serveur)

```
Subsystem sftp /usr/lib/openssh/sftp-server
```

La configuration suivante permettrait à un client d'établir une connexion à un serveur IMAP (courriels) à travers une connexion SSH sécurisée en ajustant la commande du paragraphe précédent en conséquence du nom du sous-système

```
Subsystem imap /usr/sbin/imapd
```

Exemple typique d'un fichier de configuration sshd_config

Le détail du résultat des paramètres sont situés en commentaires (suivant #) dans l'exemple.

```
# Le serveur n'acceptera que les connexions IPv4 entrantes à l'adresse IP 192.168.0.10 au port TCP 20222
#
Port 20222
AddressFamily inet
ListenAddress 192.168.0.10

# Le contenu du fichier /etc/ssh/motd sera affiché à la connexion de l'utilisateur et les tentatives d'authentification seront journalisées
#
Banner /etc/ssh/motd
SyslogFacility AUTH

# Le compte root ne peut être utilisé pour s'authentifier en SSH, l'authentification par mot de passe est permise mais si un compte ne possède pas de mot de passe, il ne pourra pas être utilisé pour se connecter en SSH
#
PermitRootLogin no
PasswordAuthentication yes
PermitEmptyPasswords no

# Seul les groupes ssh_users et ssh_serviceaccounts pourront s'authentifier en SSH
#
AllowGroups ssh_users ssh_serviceaccounts

# Par défaut, tous les utilisateurs pourront établir des tunnels réseau et afficher à distance le contenu d'une application graphique
#
AllowTcpForwarding yes
X11Forwarding yes

# Il sera possible de transférer des fichiers par SFTP et SCP2 vers et depuis le serveur
#
Subsystem sftp /usr/lib/openssh/sftp-server

# Ces paramètres plus spécifiques s'appliqueront au groupe ssh_serviceaccounts
#
Match Group ssh_serviceaccounts
  AllowTcpForwarding no
  X11Forwarding no
```

```
  # Quoi que non couvert dans le document, ce paramètre détermine le nombre de tentatives de connexions permises (la valeur divisée par deux qui est 6 par défaut) avant qu'un échec d'authentification soit journalisé et que la session soit coupée. Un compte de service étant normalement utilisé pour des tâches automatisées, aucun échec d'authentification ne devrait avoir lieu
  MaxAuthTries 2

# Ces paramètres plus spécifiques s'appliqueront à l'utilisateur "bob"
#
Match user bob
  AllowTcpForwarding yes
  X11Forwarding no
  MaxAuthTries 80
  # Cette commande sera forcée à sa connexion et sera l'unique commande permise mais l'utilisateur pourrait utiliser ce serveur pour établir un tunnel réseau à condition qu'il soit dans le groupe "ssh_users".
  ForceCommand /bin/echo 'Pas de SSH pour toi bob.'
```

Windows ?

Installation

Depuis quelques versions de Windows, il est dorénavant possible d'installer un serveur SSH et accéder à la ligne de commande avec ceci plutôt que par PowerShell Remoting. Cette fonctionnalité peut être installée/activée dans les fonctionnalités avancées/facultative mais peut aussi être simplement activée par PowerShell avec les commandes suivantes exécutées en administrateur.

```
# Trouver la version à installer
Get-WindowsCapability -Online | Where-Object Name -like "OpenSSH.Server*"
#
# Installation du serveur
Add-WindowsCapability -Online -Name OpenSSH.server~~~~0.0.1.0
# Un redémarrage sera nécessaire ici
#
# Activation du service
Set-Service -Name sshd -StartupType 'Automatic'
Start-Service -Name sshd
```

Configuration

Tout comme sous Linux, sa configuration peut être trouvée dans un fichier sshd_config situé dans C:\ProgramData\ssh\. La structure et les options du fichier sont sensiblement les mêmes que pour la configuration du serveur SSH sous Linux à quelques exceptions près telles que X11Forwarding puisque ces paramètres ne s'appliquent pas à Windows.

Il faudra par contre ajouter une règle au pare-feu pour permettre les connexions entrantes au port configuré pour SSH avec la commande suivante puisque le pare-feu Windows est activé par défaut et bloque les communications entrantes à la plupart des ports.

```
# Ajout d'une règle de pare-feu permettant les connexions entrantes au port SSH
New-NetFirewallRule -DisplayName "SSH Entrant" -Direction Inbound -LocalPort 22 -Protocol TCP
-Action Allow
```

Il est aussi nécessaire d'ajouter les utilisateurs ayant l'autorisation de se connecter en SSH au groupe correspondant. Ces commandes ajouteraient "Bob" aux utilisateurs ayant la permission de

se connecter à la machine par SSH

```
PS C:\WINDOWS\system32> Get-LocalGroup | Where-Object Name -like *SSH*

Name                 Description
----                 -----------
Utilisateurs OpenSSH Les membres de ce groupe peuvent se connecter à cet ordinateur à l’aide
de SSH.



PS C:\WINDOWS\system32> Get-LocalGroupMember -Group "Utilisateurs OpenSSH"
PS C:\WINDOWS\system32> Add-LocalGroupMember -Group "Utilisateurs OpenSSH" -Member "bob"
PS C:\WINDOWS\system32> Get-LocalGroupMember -Group "Utilisateurs OpenSSH"

ObjectClass Name                PrincipalSource
----------- ----                ---------------
Utilisateur DESKTOP-D67ERFV\Bob Local
```

Serveur

Cisco IOS ?

Support pour SSH

Il est important de noter que pour supporter HTTPS, SSH, SCP, etc., les appareils Cisco doivent exécuter une version de Cisco IOS supportant l'encryption. Il est possible d'identifier si l'encryption est supportée par le firmware par son nom. La structure des noms de firmware de Cisco est généralement composée du modèle de l'appareil, le niveau de licence (ex adventerprise pour "Advantage Enterprise" ou "lanlite" pour les fonctionnalités de base seulement), le numéro de version majeure (portant généralement un nom géographique associé tel que "Gibraltar", "Fuji", "Amsterdam", "Denali", "Everest", "Cupertino" et "Bengaluru" pour les versions 16.x et plus récentes), la révision du logiciel, le "Throttle" entre parenthèses indique la révision mineure du logiciel incluant généralement des nouvelles fonctionnalités ou quelques correctifs et le nom se termine enfin par un identifiant déterminant le type de publication ou le "train" ainsi que sa révision.

La structure suivante identifie donc le numéro de version complète et le nom complet du fichier de logiciel considérant que les fonctionnalités avancées d'entreprise ainsi que l'encryption soient inclus dans le logiciel serait "cat4500e-entservicesk9-mz.151-2.SG2.bin"

15	.1	(2)	SG	2
Major release number	Minor release number	New feature release number	Branch/train/platform identifier	Maintenance rebuild number

> TL; DR : Pour pouvoir activer toute forme de service d'encryption sur un appareil Cisco IOS, le nom du "firmware" doit contenir "k9"

Configuration du service

Lors de l'installation de OpenSSH sur la majorité des systèmes, une clé publique identifiant le système de façon unique est généralement générée automatiquement. Sur les appareils Cisco IOS, il est nécessaire de générer cette identité pour activer le service.

Afin de générer cette clé, il est impératif de configurer un nom de domaine sur l'appareil. Il faut ensuite s'assurer que le service soit activé. Si le support pour SCP est nécessaire (ex. pour prendre des sauvegardes de la configuration), il devra être activé séparément. Il faudra ensuite configurer un utilisateur avec lequel il sera possible de s'identifier (peut aussi être délégué à un serveur RADIUS) et spécifier aux terminaux virtuels d'être à l'écoute de connexions SSH ainsi que la liste d'utilisateurs à consulter pour l'authentification.

Les commandes suivantes suffisent généralement à activer SSH sur un appareil Cisco IOS.

```
Switch> enable
Switch# configure terminal

# Il est généralement recommandé de définir le nom de l'appareil puisque son identité sera générée en conséquence
Switch(config)# ip domain-name tamere.local

# Le niveau de privilège de la ligne suivante peut être ajusté, 15 est un administrateur global
Switch(config)# username bob privilege 15 password bob

# La ligne suivante affichera un avertissement lors au cours de la génération de la clé
Switch(config)# crypto key generate rsa modulus 4096

Switch(config)# ip ssh version 2
Switch(config)# ip scp server enable

# La ligne suivante peut être ajustée en fonction du nombre de terminaux à configurer pour SSH ou Telnet
Switch(config)# line vty 0 4

# La ligne suivante réserve ces terminaux virtuels pour SSH mais il est aussi possible de laisser ceux-ci à "all"
Switch(config-line)# transport input ssh

# La ligne suivante indique à ces terminaux de consulter la liste d'utilisateurs locaux pour permettre l'authentification
Switch(config-line)# login local
```

> Dans cet exemple, les terminaux virtuels 0 à 4 sont réservés à utilisation pour SSH. Il est également bonne pratique de réserver les terminaux 5 à 15 pour éviter des tentatives d'authentification par des protocoles moins sécurisés comme Telnet si possible.

Serveur

MikroTik RouterOS ??

Activation

SSH est activé par défaut sous RouterOS et une identité est générée au premier démarrage de l'appareil. Il est possible de le désactiver avec la commande "**/ip/services/set ssh disabled=yes**" et de le réactiver en changeant le paramètre "disabled" pour "no".

Configuration

Il est possible de changer son numéro de port à l'aide de la commande "**/ip/services/set ssh port=20222**". Il est possible de déterminer le nombre maximal de sessions concurrentes établies à l'aide de la commande "**/ip/services/set ssh max-sessions=5**". La commande "**/ip/services/set ssh address="192.168.0.10**" permet de définir l'adresse IP à laquelle le serveur SSH répondra. Ceci est généralement contrôlé par des règles de pare-feu dans RouterOS.

En ce qui a trait aux fonctionnalités avancées de SSH, il est possible de bloquer l'accès par mot de passe à l'aide de la commande "**/ip/ssh set always-allow-password-login=no**". Il est aussi possible d'empêcher l'établissement de tunnels réseau à l'aide de la commande "**/ip/ssh set forwarding-enabled=no**" ou permettre seulement l'établissement de lien local ou distant avec les valeurs "**local**" et "**remote**"

Client

Il est possible de configurer une panoplie d'options du côté du client aussi. Il est possible de créer des tunnels réseau, d'afficher une application graphique exécutée sur le serveur du côté du client, de renforcer la sécurité de diverses façons allant de la vérification stricte de l'identité du serveur, la limitation des méthodes de chiffrement, les algorithmes d'échange de clés ainsi que l'authentification automatisée par certificat plutôt que par nom d'utilisateur et mot de passe.

Hôtes connus, chiffrage et algorithmes

Hôtes connus

La plupart des clients SSH enregistreront les informations relatives aux serveurs connus dans un fichier nommé "known_hosts" généralement situé dans le dossier .ssh à l'intérieur du dossier d'utilisateur (ex. C:\Users\bob\.ssh\known_hosts ou /home/bob/.ssh/known_hosts).

La structure de ce fichier est que chaque ligne est composée de trois informations séparées par un espace. La première information est l'adresse du serveur (que ce soit son adresse IP ou un nom de domaine), la deuxième information est la méthode de chiffrement négociée entre le client et le serveur et la troisième information est la signature/identité du serveur.

À la première connexion à un serveur, son identité nous est présentée et le client négocie avec le serveur un algorithme d'échange de clés et une méthode de chiffrement de la connexion. Ces informations sont alors stockées et en cas de changement de ces informations, le client avertira que l'identité du serveur a changé. Si vous n'avez pas vous-même changé l'identité du serveur, il est possible qu'un acteur malveillant tente de se faire passer pour le serveur par une attaque de type "homme-au-milieu" ou "man-in-the-middle" (MITM).

Configuration du client

Certains clients considèrent certains algorithmes ou méthodes de chiffrement comme étant désuets et les désactivera par défaut. Le fichier "config" ou "ssh_config" situé dans le même dossier que le fichier "known_hosts" permet de spécifier des paramètres supplémentaires à la connexion à un serveur spécifique ou à plusieurs serveurs.

Ligne de commande

La méthode prédominante et celle utilisées à fin d'automatisation est évidemment l'outil SSH à la ligne de commande. Il est aussi possible à l'aide de librairies telles que netmiko ou paramiko d'interagir avec un serveur SSH en Python pour des besoins spécifiques mais généralement, l'ensemble des manipulations qui pourraient être nécessaires peuvent être effectuées à partir d'un shell.

Installation

Linux

Dépendament de la distribution de Linux que vous utiliserez, le gestionnaire de packages aura une structure différente mais le nom du package en question est généralement "ssh".

Voici quelques exemples.

```
# Debian/Ubuntu
sudo apt update && sudo apt install ssh

# RHEL/Fedora
sudo dnf install ssh

# ArchLinux
sudo pacman -S ssh
```

Un autre outil essentiel à l'automatisation de tâches par SSH est la commande "expect". Elle permet d'automatiser l'exécution de commandes complexes à travers une connexion ssh.

```
# Debian/Ubuntu
sudo apt update && sudo apt install expect

# RHEL/Fedora
sudo dnf install -y tcl tk expect

# ArchLinux
sudo pacman -S expect
```

Windows

Le client SSH sous Windows est une fonctionnalité facultative pouvant être installée en PowerShell.

```
# Exécuter en tant qu'administrateur
Add-WindowsCapability -Online -Name OpenSSH.Client~~~~0.0.1.0
```

Utilisation

Connexion simple par utilisateur et mot de passe

Si le serveur SSH le permet, la commande suivante est généralement suffisante pour établir une connexion au shell du serveur distant. Le domaine est optionnel, il sera généralement utilisé pour se connecter à un serveur Windows si celui-ci fait partie d'un domaine Active Directory.

```
ssh [domaine\utilisateur]@[adresse du serveur] -p [port du serveur distant]
```

Connexion par certificat

Si le serveur SSH le permet, les commandes suivantes permettent de créer une clé SSH publique permettant l'authentification sans mot de passe. Ceci est généralement essentiel lors d'automatisation de tâches. Il est à noter que ces commandes doivent être exécutées par l'utilisateur qui s'authentifiera au serveur. Celle-ci n'est pas pour l'ensemble du système.

```
# Génération du certificat, le certificat peut être placé ailleurs mais son emplacement par défaut est généralement dans le dossier .ssh de l'utilisateur et porte généralement le nom id_rsa. Ce chemin est ici mentionné sous la forme pour Linux.
ssh-keygen -f ~/.ssh/id_rsa -N [mot de passe, au besoin, sera demandé au moment de la connexion] -t rsa -q

# Copie de la clé publique sur un serveur SSH auquel on pourra s'authentifier par certificat
ssh-copy-id -i [chemin du fichier] [domaine\utilisateur]@[adresse du serveur] -p [port du serveur SSH]

# À ce moment, la connexion SSH ne devrait plus demander de mot de passe
ssh [domaine\utilisateur]@[adresse du serveur] -p [port du serveur SSH]
```

Exécution d'une commande individuelle sans réponse

Il est possible d'exécuter une seule commande de façon non interactive avec l'argument "-c"

```
# La commande suivante permettrait d'automatiser la mise à jour d'un système Debian/Ubuntu si sudo est configuré pour laisser l'utilisateur exécuter des commandes avec sudo sans demander de mot de passe
ssh [domaine\utilisateur]@[adresse du serveur] -p [port du serveur] -c "sudo apt update && apt upgrade"

# La commande suivante permettrait de redémarrer le serveur distant
ssh [domaine\utilisateur]@[adresse du serveur] -p [port du serveur] -c "sudo reboot"
```

Partage de ressources réseau/établissement de tunnels

Si le paramètre "AllowTcpForwarding" est défini à "yes" dans la configuration du serveur, il est possible d'établir des tunnels pour mettre à disposition du client ou du serveur des ressources réseau se situant d'un côté ou de l'autre. Un autre livre décrit ce concept plus en profondeur.

```
# La commande suivante metterait à disposition du client un serveur MySQL exécuté sur le serveur SSH comme s'il était exécuté sur le client. Ceci est le "remote forwarding" où on prend un port du serveur SSH et on le met à la disposition du client.
ssh [domaine\utilisateur]@[adresse du serveur] -p [port du serveur] -R 3306:localhost:3306

# La commande suivante metterait à disposition du client un serveur web auquel le serveur SSH aurait accès comme si le serveur était exécuté sur le client. Ceci est aussi une forme de "remote forwarding"
ssh [domaine\utilisateur]@[adresse du serveur] -p [port du serveur] -R 443:[adresse du serveur web, ex. www.google.com]:443

# La commande suivante metterait à disposition du serveur un serveur RDP (bureau à distance) exécuté localement sur le client. Ceci est une forme de "local forwarding" et est l'inverse du remote
ssh [domaine\utilisateur]@[adresse du serveur] -p [port du serveur] -L 3389:localhost:3389

# La commande suivante metterait à disposition du serveur un serveur SSH auquel le client a accès comme par exemple le routeur du client. Ceci est une forme de "local forwarding" et est l'inverse du remote. Il est à noter ici que le premier port mentionné est le port qui sera utilisé sur le serveur distant pour se connecter et doit donc différer de son propre port SSH. Ce genre de connexion permettrait une porte d'entrée dans un réseau du client au serveur distant. Ceci est nommé un "backdoor".
ssh [domaine\utilisateur]@[adresse du serveur] -p [port du serveur] -L 8022:[adresse du routeur]:22
```

```
# La commande suivante établirait une redirection de ports dynamiques qui pourrait être utilisé à titre de proxy SOCKS. Ceci porte aussi le nom d'un "poor man's VPN". On pourrait donc ici configurer un proxy pointant vers localhost au port 8443 et le trafic traverserait la connexion SSH et sortirait du serveur SSH.
ssh [domaine\utilisateur]@[adresse du serveur] -p [port du serveur] -D localhost:8443
```

Transfert de fichiers

SSH permet généralement le transfert de ficheirs à l'aide du protocole "SCP". Les fichiers peuvent être récupérés ainsi que déposés sur le serveur. Il est à noter que le compte du serveur auquel on se connecte doit posséder les autorisations nécessaires pour effectuer l'opération demandée.

```
# La structure de SCP est la suivante. Il est à noter qu'avec cette structure, il est possible de transférer un fichier d'un serveur SSH à un autre. Si aucun utilisateur et adresse sont mentionnés d'un côté ou de l'autre et que seulement le chemin complet du fichier (source ou destination) est mentionné, il s'agit alors d'un fichier situé sur le stockage du client.
scp [domaine\utilisateur]@[adresse de la source]:[chemin complet du fichier source] [domaine\utilisateur]@[adresse de la destination]:[chemin complet du fichier source]

# La commande suivante permettrait de récupérer une copie du journal système d'un serveur distant
scp root@[adresse du serveur]:/var/log/syslog [chemin de la destination]

# La commande suivante permettrait de déposer un fichier de configuration bash dans le dossier de l'utilisateur distant
scp [chemin du fichier de configuration] [utilisateur]@[serveur]:/home/[utilisateur]/.bashrc
```

Exécution interactive de commandes avec "expect"

La commande "expect" comme son nom l'indique fonctionne en réagissant au résultat des commandes exécutées. Il utilise des expressions régulières pour valider qu'il est le bon moment pour exécuter certaines commandes. Il peut être utilisé tant à des fins d'exécutions locales qu'à l'exécution de commandes distantes à travers une connexion SSH.

Le script suivant permettrait de s'authentifier en SSH à l'aide d'un mot de passe automatiquement et d'élevé l'invite de commande à l'utilisateur "root".

```
#!/usr/bin/expect

# Utilisation ssh_switchuser.expect <host> <ssh user> <ssh password> <su password>

set timeout 60
```

```
# Établissement de la connexion SSH
spawn ssh [lindex $argv 1]@[lindex $argv 0]

# S'il s'agit de la première connexion, accepter l'identité du serveur SSH
expect "yes/no" {
send "yes\r"
    # Si le serveur demande un mot de passe, mentionner le mot de passe
expect "*?assword" { send "[lindex $argv 2]\r" }
} "*?assword" { send "[lindex $argv 2]\r" }

# Exécution de la commande d'élévation à root
expect "# " { send "su - \r" }
# Si un mot de passe est demandé, mentionner le mot de passe
expect ": " { send "[lindex $argv 4]\r" }
expect "# " { send "ls -ltr\r" }

# Retourner le contrôle à l'utilisateur dans l'état actuel de l'invite de commande, ici dans une connexion SSH établie
interact
```

Ce script aurait pu continer pour effectuer d'autres actions automatiquement et fermer la connexion SSH suite à ces actions ainsi que d'écrire le résultat des commandes dans un fichier.