|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SSH C:\Users\LSAVARD\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\F2D9F6A2.tmp  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  SSH (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png |  | Communications sécurisées **Outils de chiffrement et de signature openSSH**  **Introduction**  *Secure SHell* (SSH) est à la fois un programme informatique et un protocole de communication sécurisé.  Le protocole de connexion impose un échange de clés de chiffrement en début de connexion. Par la suite, tous les segments TCP sont authentifiés et chiffrés. Il devient alors impossible d'utiliser un renifleur (*sniffer*) pour voir ce que fait un utilisateur.  Le protocole SSH a été conçu avec l'objectif de remplacer les différents protocoles non chiffrés comme *rlogin*, *telnet*, *rcp* et *rsh*.  **Protocole SSH**  Le protocole SSH existe en deux versions majeures : la version 1.0 et la version 2.0.   * La première version permet de se connecter à distance à un ordinateur afin d'obtenir un *shell* ou ligne de commande.  Cette version souffrait néanmoins de problèmes de sécurité dans la vérification de l'intégrité des données envoyées ou reçues, la rendant vulnérable à des attaques actives.  En outre, cette version implémentait un système sommaire de transmission de fichiers, et du port *tunneling*. * La version 2 est beaucoup plus sûre au niveau cryptographique, et possède en plus un protocole de transfert de fichiers complet, le *SSH file transfer protocol* (SFTP).   Habituellement le protocole SSH utilise le port TCP 22.  Il est particulièrement utilisé pour ouvrir un *shell* sur un hôte distant.  Peu utilisé sur les stations Windows (quoiqu'on puisse l'utiliser avec *PuTTY*, *mRemote*, *cygwin* ou encore *openSSH*), SSH fait référence pour l'accès distant sur les stations Linux et Unix.  SSH peut également être utilisé pour transférer des ports TCP d'un hôte vers un autre, créant ainsi un tunnel.  Cette méthode est couramment utilisée afin de sécuriser une connexion qui ne l'est pas (par exemple le protocole de récupérations de courrier électronique POP3) en la faisant transférer par le biais du tunnel chiffré SSH.  Il est également possible de faire plusieurs sauts entre consoles SSH, c'est-à-dire ouvrir une console sur un serveur, puis, de là, en ouvrir une autre sur un autre serveur.  Les usages de base de SSH sont ...   * accéder à distance à la console en ligne commande (shell), ce qui permet, entre autres, d'effectuer la totalité des opérations courantes et/ou d'administration sur la machine distante ; * déporter l'affichage graphique de l’hôte distant ; * transferer des fichiers en ligne de commande ; * monter ponctuellement un répertoire distants, soit en ligne de commande, soit via un interface graphique ; * monter automatiquement des répertoires distants.   **SSH avec authentification par clés**  Avec SSH, l'authentification peut se faire sans l'utilisation de mot de passe ou de phrase secrète en utilisant le chiffrement asymétrique.   * La clé publique est distribuée sur les systèmes auxquels on souhaite se connecter ; * La clé privée, qu'on prendra le soin de protéger par un mot de passe, reste uniquement sur le poste de travail à partir duquel on se connecte.  L'utilisation d'un agent SSH (*ssh-agent*) permet d'emmagasiner le mot de passe de la clé privée pendant la durée de la session utilisateur.   Cette configuration profite aussi à SCP et à SFTP qui se connectent au même serveur SSH.  **Tunnel SSH**  Un tunnel, dans le contexte de réseaux informatiques, est une encapsulation de données d'un protocole réseau dans un autre, situé dans la même couche du modèle en couches, ou dans une couche de niveau supérieur.  Par exemple, pour faire passer le protocole IPv6 par le réseau Internet actuel (qui est en grande partie en IPv4) on va créer un tunnel entre deux machines IPv4 ; ce tunnel, pour le protocole IPv6, semblera un simple lien point-à-point  (un logiciel comme *traceroute* ne verra donc pas le tunnel).  En sécurité, on crée souvent des tunnels chiffrés, par exemple comme le fait SSH.  Les données peuvent alors y circuler sans craindre d'être écoutées. Les tunnels peuvent être utilisés pour créer des réseaux privés virtuels (VPN).  Le tunnel HTTP est un cas particulier qui consiste à faire passer une connexion explicitement interdite par un pare-feu (le protocole SSH par exemple) encapsulé dans un protocole HTTP presque toujours autorisé. C'est particulièrement utile quand le pare-feu ne bloque pas simplement les ports associés (22 pour SSH par exemple), mais analyse aussi les protocoles utilisés.  **openSSH**  La suite logicielle OpenSSH inclut les outils suivants ...   * **ssh** Remplaçant pour les clients rlogin et telnet ... # **ssh** [**utilisateur@profsavard.info**](mailto:utilisateur@profsavard.info) * **scp** Remplaçant pour le client rcp ... # **scp utilisateur@profsavard.info:~/tux .** * **sftp** Remplaçant pour le client ftp ... # **sftp utilisateur@profsavard.info** * **sshd** Démon (*daemon*) SSH ... # **sshd ...** * **sftp-server** Démon (*daemon*) remplaçant le serveur FTP. sftp-server est un sous-système lancé par sshd quand celui-ci reçoit une demande de connexion d'un client sftp. # **sftp-server ...** * **ssh-keygen** Programme de génération, gestion et conversion des clés RSA, DSA et DSA.  Par défaut, la clé privée générée par *ssh-keygen* est un fichier texte sans extension (dont l'utilisation peut être protégé par une phrase secrète) et la clé publique associée est un fichier texte d'extension *.pub*.  Ces paires de clés servent soit à l'authentification des hôtes (attribut *HostKey* du serveur *sshd*) soit à l'authentification des utilisateurs (attribut *IdentityFile* du client *ssh*). # **keygen -t ecdsa -b 384 -f .ssh/<Clef> -C <commentaire>** * **ssh-agent** Agent d'authentification. * **ssh-add** Agent de gestion des clés privées de l'utilisateur. *ssh-agent* et *ssh-add* font office de trousseau de clés. * **ssh-keyscan** Utilitaire de récupération et vérification des clés publiques d'hôtes distants. * **ssh-keysign** Agent d'authentification servant si l'option *HostbasedAuthentication* est activée sur le serveur (option désactivée par défaut). * **ssh-copy-id** Utilitaire servant à déposer sa clé publique sur un serveur distant.   openSSH utilise le chiffrement asymétrique comme mécanisme d'authentification.  À ce titre, OpenSSH gère les clés RSA, les clés DSA et les clés DSA basées sur les courbes elliptiques.  OpenSSH reconnaît aussi les certificats X509 et les fichiers au format PKCS#12.  L'installation du serveur oenSSH crée un dossier ***/etc/ssh*** et génère un couple de clés RSA …   * ***/etc/ssh/ssh\_host\_rsa\_key*** contenant la clé privée du serveur ; * ***/etc/ssh/ssh\_host\_rsa\_key.pub*** contenant la clé publique du serveur.   Lorsqu'un client demande une connexion au serveur, ce dernier envoie au client sa clé publique. Le client utilise alors la clé publique reçue pour envoyer au serveur une clé secrète. Le serveur reçoit alors la clé publique et la décrypte avec sa clé privée, prouvant ainsi qu'il est bien le serveur.  Pour le prouver au client, le serveur crypte un message type avec la clé publique du client et lui envoie. Si le client arrive à lire le message en le déchiffrant avec sa clé privée, il a l'assurance de communiquer avec le véritable serveur. Le canal de communication sécurisée est alors créé.  **Modèle de sécurité**  Le modèle de sécurité d'openSSH n'utilise pas de PKI/IGC (*Public Key Infrastructure* ou Infrastructure de Gestion des Clés) mais préfère la méthode du *Trust on first use* ; avec cette méthode, chaque nouvelle connexion à un serveur SSH jusque-là inconnu du client nécessite l'approbation de l'utilisateur.  Si l'utilisateur accepte la clé publique que lui présente ce serveur SSH (soit sans vérification ce qui n'est pas souhaitable, soit après vérification de l'empreinte de la clé publique que l'utilisateur aura obtenue auparavant) le client ssh poursuivra les autres étapes de la connexion SSH pendant que l'utilitaire *ssh-keyscan* enregistrera la clé publique de l'hôte dans un fichier nommé *known\_hosts*. Le fichier *known\_hosts* conserve l'ensemble des clés publiques associées aux hôtes (serveurs SSH) qui ont été approuvées.  Pour toute nouvelle connexion auprès d'un hôte (serveur SSH) connu, l'utilitaire *ssh-keyscan* va comparer l'empreinte de la clé publique que lui soumet le serveur avec celle qu'il a précédemment enregistrée dans le fichier *known\_hosts*.  Un exemple … Disons que vous rencontrez quelqu'un pour échanger des secrets d'affaires. Votre conseiller vous dit que vous n’avez jamais rencontré cette personne auparavant et que cela peut être un imposteur. Lors des prochaines réunions avec cette personne, votre conseiller ne vous préviendra plus.  Lors du branchement sur hôte distant, le message suivant apparaît … # **ssh -T** [**tux@192.168.1.11**](mailto:tux@192.168.1.11) ***The authenticity of host '192.168.1.11 (192.168.1.11)' can't be established. RSA key fingerprint is SHA256:1CZKHORTXpZtEvFJDzyZPSK68Hqe9DzBjjZlE2zbiew. Are you sure you want to continue connecting (yes/no)?***  C'est ce que le message signifie. La personne est le serveur distant et votre conseiller est le client ssh.  Il peut être nécessaire de vérifier l'identité de la personne avant de partager des secrets avec elle.  La commande suivante permet de vérifier le tout … # **ssh-keyscan -t rsa <Adresse IP> | ssh-keygen -lv -f -** # **ssh-keyscan -t rsa srv01.profsavard.info | ssh-keygen -lv -f – # 192.168.1.11:22 SSH-2.0-OpenSSH\_7.4p1 Debian-10+deb9u6 2048 SHA256:1CZKHORTXpZtEvFJDzyZPSK68Hqe9DzBjjZlE2zbiew 192.168.1.11 (RSA) +---[RSA 2048]----+ | .o . =\*o+ | | o + ++oO+o | | = +oo+oo.. | | ..+.o+ | | .oS= = . | | o X o | | ..\* o | | ..=+E | | +o.o. | +----[SHA256]-----+**  Si les deux correspondent, il est préférable d’ajouter la clé au fichier ~/.ssh/known\_hosts  (emplacement standard dans de nombreuses distributions Linux) avec … # **ssh-keyscan -t rsa -H srv01.profsavard.info >> ~ / .ssh / known\_hosts** et le client ssh n’affichera plus le message de sécurité. Il comparera les clefs à chaque fois que l’on se connectera.  Dans le cas d'un imposteur (par exemple, une attaque de type homme au milieu), le client ssh rejettera la connexion car la clef aura changée.  **Configuration du serveur d'openSSH**  **Vérifications de base** Pour accéder à un hôte (tout noeud branché sur le réseau Internet - un ordinateur personnel, un serveur local, un serveur distant dont on effectue l'administration, ...), il faudra installer le paquet ***openssh-server*** sur l'hôte à rejoindre en SSH. Cet hôte sera le serveur SSH.  La partie cliente est fournie par le paquet ***openssh-client***, qui est installé par défaut sur la plupart des distributions Linux.  Pour vérifier ce qui est déjà installé ... # **ssh -V *OpenSSH\_7.4p1 Debian-10+deb9u6, OpenSSL 1.0.2r 26 Feb 2019*** et  la commande dpkg afin de connaître la version de la bibliothèque ssl ... # **dpkg -l libssl\* *Souhait=inconnU/Installé/suppRimé/Purgé/H=à garder | État=Non/Installé/fichier-Config/dépaqUeté/échec-conFig/H=semi-installé/W=attend-traitement-déclenchements |/ Err?=(aucune)/besoin Réinstallation (État,Err: majuscule=mauvais) ||/ Nom Version Architecture Description +++-==============-============-============-================================= ii libssl1.0.2:am 1.0.2r-1~deb amd64 Secure Sockets Layer toolkit – sh ii libssl1.1:amd6 1.1.0j-1~deb amd64 Secure Sockets Layer toolkit – sh***  **Utilisation du serveur SSH**  Le serveur SSH fonctionne en tant que service lancé automatiquement au démarrage de la machine.  Il est possible notamment de ...   * l'activer ou l'arrêter si on souhaite désactiver momentanément le serveur SSH ;à * le relancer si on fait une modification de configuration   **Activation du service ...** Pour le serveur # **systemctl start ssh** Pour le client # **systemctl start sshd**  **Arrêt du service ...** Pour le serveur **# systemctl stop ssh** Pour le client # **systemctl stop sshd**  **Relance du service (redémarrage) ...** Pour le serveur # **systemctl restart ssh** Pour le client # **systemctl restart sshd**  **Utilisations de SSH**  **Accès à distance à la console en ligne de commande (*shell* ssh)**  Pour ouvrir une session distante ayant un serveur SSH, on doit saisir la commande suivante ... # **ssh <Nom utilisateur>@<Adresse IPv4> -p <Port>** # **ssh tux@srv01.profsavard.info -p 12345**  L'option -p <Port>, qui précise le port utilisé par le serveur, est facultative.  Si rien n'est précisé, c'est le port TCP 22 qui sera utilisé par défaut.  Pour se connecter avec SSH en IPV6 depuis un terminal ... # **ssh -6 <Nom utilisateur>@<Adresse ipv6>**  # **ssh -6 tux@2a01:e35:2431::2e57**  **Remarque** ... Afin de pouvoir se connecter en IPV6, il faut que le serveur SSH puisse écouter les adresses IPV6.  Pour ce faire, il faut ajouter le code suivant dans le fichier ***/etc/ssh/sshd\_config*** sur le serveur ... **ListenAddress ::**  Il est aussi possible d'appeler un hôte par son nom ... # **ssh tux@srv01.profsavard.info**  **Astuces** ... Afin d'accélérer la résolution des noms d'hôtes, il est intéressant d'ajouter, au fichier */etc/hosts*, les noms d'hôtes (et adresses ip) des serveurs les plus utilisés.  Si on souhaite se connecter à plusieurs hôtes situés derrière un routeur, il est possible de configurer ce dernier afin qu'il redirige chaque port TCP entrant vers un hôte donné ...   * port externe 22001 redirigé vers 192.168.1.1:22 ; * port externe 22002 redirigé vers 192.168.1.2:22.   Il faudra utiliser l'option -p 22001 pour se connecter sur l'hôte ayant pour adresse 192.168.1.2 sur le réseau local.  **Authentification**  **Authentification par mot de passe** L'authentification par mot de passe (transmis chiffré) est le mode d'identification par défaut. Suite à l'installation du paquet *openssh-server*, il peut parfois être nécessaire de modifier le fichier de configuration ***/etc/ssh/sshd\_config***, notamment si on rencontre le problème suivant ... # **ssh** [**tux@srv01.profsavard.info**](mailto:tux@srv01.profsavard.info) ***Permission denied (publickey).***  Dans ce cas, il faut très simplement modifier avec les droits d'administration le fichier ***/etc/ssh/sshd\_config*** sur le serveur SSH ... **# Change to yes to enable tunnelled clear text passwords PasswordAuthentication yes**  Suite à cette modification il faudra relancer le service.  **Attention** ... Si on expose son serveur SSH sur Internet, ne JAMAIS oublier qu'Internet est parcouru en permanence par des robots qui balayent (*scan*) et testent en permanence tous les serveurs (SSH et autres) et qu'ils vont faire des tentatives pour trouver les mots de passe de compte.  L'usage des clés est donc fortement recommandé. Sinon, il est préférable d'utiliser des mots de passe longs et complexes ou encore utiliser des utilitaires de protection comme ***fail2ban*** qui permet de bannir des adresses IP.  **Authentification par chiffrement asymétrique** Autrefois, tout le monde employait l'authentification typique par le principe identifiant – mot de passe.  Cependant si quelqu'un connaît ce mot de passe, la sécurité est compromise.  Pour adresser le problème, SSH offre l'Authentification par clés publique/privée au lieu des simples mots de passe.  De cette manière, il faut être en possession de non plus d'une mais de deux informations pour se connecter  (avoir la clé privée et connaître le mot de passe de cette clé).  Ceci peut permettre, par exemple, ...   * à un administrateur de se connecter à plusieurs hôtes sans devoir connaître des centaines de mots de passe différents ; * de ne pas avoir un mot de passe à saisir toutes les 2 minutes (en utilisant *ssh-agent*).   À moins que l'on ait déjà une paire de clés, il faut d'abord en créer une.  Pour une **clé utilisant le protocole de chiffrement RSA** ... # **ssh-keygen -t rsa** ***Enter file in which to save the key (/root/.ssh/id\_rsa):***  openSSH demande où sauvegarder la clé privée  (on accepte l'endroit par défaut soit ***~/.ssh***, et on ne modifie pas le nom du fichier généré). Il faudra, par mesure de sécurité, choisir une phrase de reconnaissance (*passphrase*) … ***Enter passphrase (empty for no passphrase): Enter same passphrase again: Your identification has been saved in /root/.ssh/id\_rsa. Your public key has been saved in /root/.ssh/id\_rsa.pub. The key fingerprint is: SHA256:3kuyyLoWBwMDxvXVFQ1hNA9I71p6GUphVi0GViui7kA*** [***tux@srv01.profsavard.info***](mailto:root@srv01.profsavard.info) ***The key's randomart image is: +---[RSA 2048]----+ |oo.. .o+\*%\* | |..o . . .o+o=o | | o . . =.o.. | | o . + + | | E + S. + | | . o ....= o | | . + o+oo | | +. . +.. |***  ***| .o+o . . | +----[SHA256]-----+***  **Remarque** ... Bien que cela ne soit pas obligatoire, l'utilisation d'une phrase de reconnaissance est recommandée pour protéger sa clé privée. En effet, toute personne qui obtiendrait l'accès à la clé privée (non protégée) aurait alors toutes les permissions sur d'autres hôtes.  Il faut prendre un moment et choisir une bonne phrase de reconnaissance, longue et complexe.  La clef publique sera créée avec la nouvelle clef privée.  Elles sont habituellement localisées dans le répertoire caché ***~/.ssh*** ...   * ***~/.ssh/id\_rsa*** pour la clé privée ; et * ***~/.ssh/id\_rsa.pub*** pour la clé publique.   Il faut maintenant envoyer au serveur la clé publique pour qu'il puisse chiffrer des messages.  En résumé ...   * La clé publique du client doit se trouver dans le fichier $***HOME/.ssh/authorized\_keys*** du serveur ; * Il faut que le client ait mis sa clé privée en ***$HOME/.ssh/*** (côté client) ; * Le répertoire ***$HOME/.ssh*** doit appartenir (*chown*) au propriétaire de *$HOME* et être en protection *700* (interdit aux autres) ; * Sur le serveur SSH, il vaut mieux refuser l'accès par mot de passe (*PasswordAuthentication no* dans */etc/ssh/sshd\_config* du serveur).   L'utilisateur distant doit avoir cette clé (c'est une ligne de caractères en code ASCII) dans son fichier de clés d'autorisation situé à ***~/.ssh/authorized\_keys*** sur le système distant.  Pour ce faire, on emploie la commande ***ssh-copy-id***. ***ssh-copy-id*** est un script qui utilise *ssh* pour se connecter à un hôte à distance en utilisant le mot de passe de l'utilisateur. L'authentification par mot de passe doit donc être autorisée dans le fichier de configuration du serveur ssh. Il modifie également les permissions des répertoires ***~/.ssh*** et ***~/.ssh/authorized\_keys*** de l'hôte distant pour retirer l'accès en écriture du groupe (ce qui empêcherait de se connecter si le serveur distant *ssh* a comme paramètre ***StrictModes yes*** dans son fichier de configuration. # **ssh-copy-id -i ~/.ssh/id\_rsa.pub <Nom utilisateur>@<adresse IP>** ou si le port est différent du port standard 22 (attention aux guillemets) # **ssh-copy-id -i ~/.ssh/id\_rsa.pub -p <Port> "<Nom utilisateur>@<Adresse Ip>"** # **ssh-copy-id -i ~/.ssh/id\_rsa.pub** [**tux@srv01.profsavard.info**](mailto:tux@srv01.profsavard.info) ***/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/tux/.ssh/id\_rsa.pub" The authenticity of host '192.168.1.230 (192.168.1.230)' can't be established. ECDSA key fingerprint is SHA256:YiOem/bZcIvG/2NMXvv2Gx0YvxXqx/ra2qKV4YD/meg. Are you sure you want to continue connecting (yes/no)? yes /usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed /usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys tux@192.168.1.230's password: Number of key(s) added: 1***  On peut maintenant essayer de se connecter à la machine avec ***ssh srv01.profsavard.info*** et s’assurer que seules les clés souhaitées ont été ajoutées. # **ssh tux@srv01.profsavard.info *Enter passphrase for key '/home/tux/.ssh/id\_rsa': Linux srv01.profsavard.info 4.9.0-9-amd64 #1 SMP Debian 4.9.168-1+deb9u2 (2019-05-13) x86\_64  The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/\*/copyright.***  ***Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.***  ***Last login: Fri May 31 09:18:15 2019 from 192.168.1.230***  **Remarque** ... L'hôte distant demande à nouveau un mot de passe ... ***Enter passphrase for key '/home/lsavard/.ssh/id\_rsa':*** Il s'agit du mot de passe de la clé et non celle de l'usager.  Pour éviter la demande à chaque connexion, il faudra configurer l’utilitaire ***ssh-agent***.  Astuce ... Si l'hôte distant refuse la connexion, il faut modifier le fichier de configuration du serveur SSH ... # **nano /etc/ssh/sshd\_config** ou # **vim /etc/ssh/sshd\_config**  Il faut décommenter ou modifier l'entrée ... ***PasswordAuthentication no*** pour **PasswordAuthentication yes** et **PubkeyAuthentication yes**  Il faut par la suite redémarrer le service sshd ... # **systemctl restart sshd.service**  Dorénavant, on a plus à utiliser le mot de passe mais la phrase de reconnaissance pour se connecter.  Celle-ci sert à déchiffrer la clé privée du système local.  **Astuce** … En cas de difficulté, c'est-à-dire que le mot de passe est toujours demandé, le suivi des connexions peut se faire à l'aide de la commande tail -f ... # **tail -f /var/log/auth.log** ou # **journalctl -f** lors des tentatives de connexion.  **Attention** ... Si le système distant retourne le mot ***vulnkey*** dans son message d'erreur, *ssh-keygen* a probablement généré une clé vulnérable.  Il faudra recommencer la procédure depuis le début (depuis la création des clefs).  Pour résumer, deux choses sont nécessaires pour obtenir un accès réellement sécurisé : par authentification à clé publique par rapport à l'authentification par mot de passe classique ...   * sa clé privée, chiffrée ; * sa phrase de reconnaissance (*passphrase*), utilisée pour déchiffrer sa clé privée.   Si on choisit de ne pas avoir de mot de passe (ce qui sera possible dans la section suivante), on aura une sécurité moindre, ainsi que si on utilise une authentification uniquement par mot de passe, comparé à celle que l'on peut avoir en combinant les deux.  **Éléments importants en lien avec l'usage des clés**  **Authentification par mot de passe et/ou par clé** Il est possible avec SSH d'avoir les deux modes d'authentifications actifs en même temps, par mot de passe et par clés : on peut vouloir neutraliser l'authentification par mot de passe pour des raisons de sécurité, pour cela il faut modifier le fichier de configuration ***/etc/ssh/sshd\_config*** de la manière suivante ...   * à la ligne ***PasswordAuthentication*** indiquer ***no*** ; * à la ligne ***UsePAM*** indiquer ***no***.   Avec ***ChallengeResponseAuthentication*** et ***PasswordAuthentication*** à ***no*** on peut continuer à utiliser PAM en bloquant l'usage des mots de passe pour SSH.  **Remarque** … Ne pas oublier de relancer le service ssh sur le serveur après avoir modifié la configuration.  **Mot de passe toujours demandés avec authentification par clés** Si, après avoir suivi ce qui précède, un mot de passe est toujours demandé, il se peut que ce soit dû à un problème de droits sur le répertoire personnel.  Sur l'hôte distant, il faut jeter un coup d'œil sur le fichier ***/var/log/auth.log*** pour y trouver des indications et notamment si la ligne suivante apparaît ... ***Authentication refused: bad ownership or modes for directory /home/votre\_login***  Il faut alors modifier les permissions ... # **chmod 755 $HOME**  Le tout devrait rentrer dans l'ordre. Si ce n'est toujours pas le cas, c'est que le serveur doit être configuré en mode de sécurité strict. Pour ce faire, il faut effectuer les opérations suivantes ...  Sur le serveur ... Dans le fichier ***/etc/ssh/sshd\_config***, la ligne ***StrictModes*** ***yes*** indique que le serveur va être très pointilleux sur les droits du compte sur lequel on se connecte en SSH. Il faut saisir ensuite les commandes suivantes ... # **chmod go-w ~/** # **chmod 700 ~/.ssh** $ **chmod 600 ~/.ssh/authorized\_keys** Sur le client, dans le fichier /etc/ssh/ssh\_config, on doit rajouter la ligne ...  **PreferredAuthentications publickey**.  **Utilisation de ssh-agent**  *ssh-agent* est un autre outil développé par openSSH permettant d'utiliser l'authentification par échange de clés avec une phrase de reconnaissance (*passphrase*) sans avoir à toujours fournir cette dernière. Une fois *ssh-agent* démarré, toutes les commandes ou programmes exécutés par SSH sont des sous-processus de *ssh‑agent*, ne nécessitant plus d'authentification ultérieure.  Ainsi, il ne reste donc plus qu'à récupérer les variables d'environnement, de les mettre en place pour *ssh-agent* et demander à *ssh-agent* de s'occuper de sa clé publique et le tour est joué...  # **ssh-agent**  ***SSH\_AUTH\_SOCK=/tmp/ssh-5ha6FG2vmSmL/agent.7893; export SSH\_AUTH\_SOCK; SSH\_AGENT\_PID=7894; export SSH\_AGENT\_PID; echo Agent pid 7894;***  Les variables d'environnement s'affichent à l'écran.  Pour ajouter une clef vers ***ssh-agent*** … # **ssh-add ~/.ssh/id\_dsa.pub** ou # **ssh-add id\_rsa.pub**  **Remarque** ... ***ssh-agent*** n'est effectif que dans la console dans laquelle il a été lancé. Si on ferme la console et en ouvre une autre, il faut tout recommencer.  **Message d'erreur** (*ssh-agent*)  Open peut afficher le message suivant … ***Could not open a connection to your authentication agent.***  La première raison de ce message est le processus *ssh-agent* lui-même.  Ce processus intercepte la clé publique lorsque le programme ***ssh-add*** reçoit des clés à ajouter.  Il faut donc que ce processus soit lancé avant d'utiliser la commande ***ssh-add***.  # **eval $(ssh-agent) *Agent pid 7897*** ou # **eval `ssh-agent -s`** ***Agent pid 7897***  Il est possible que l’on doive démarrer le service … # **exec ssh-agent bash** ou # **eval `ssh-agent`**  ***Agent pid 7897***  **Transfert - copie de fichiers** Pour copier un fichier à partir d'un hôte vers un autre à l'aide de SSH, il faudra utiliser la commande *scp* ou *rcp*.  Cela ressemblera à ceci ... # **scp <Fichier> <Utilisateur>@<Adresse IPv4 distante>:<Répertoire de destination>** et en IPv6 # **scp -6 <Fichier> <Utilisateur>@[Addresse IPv6 distante>:<Répertoire de destination>**  Pour un fichier ... # **scp fichier.texte** [**tux@192.168.1.103:/home/tux**](mailto:tux@192.168.1.103:/home/tux) et en IPv6 # **scp -6 fichier.texte tux@[2a01:e35:2431::2a34]:/home/tux**  Pour un répertoire ... # **scp -r /home/tux tux@192.168.1.103:/home/tux/** et en IPv6 # **scp -6r /home/tux tux@[2a01:e35:2431::2a34]:/home/tux**  Il est aussi possible de copier des fichiers à partir des hôtes à distance sur le disque local ... # **scp tux@192.168.1.103:/home/tux/rapport.texte .**  **Attention** ... Le point (.) à la fin de la commande précédente indique de copier le fichier dans le répertoire courant.  **Pour les noms de fichiers ou répertoires contenant des espaces ...** # **scp tux@srv01.profsavard.info:"le\ fichier" .** # **scp tux@srv01.profsavard.info:'"le fichier"' .**  Notez les simples ' ET doubles " guillemets ' " … # **fichier="le fichier"**  ou # **fichier=le\ fichier**  # **scp tux@srv01.profsavard.info:"'$fichier'" .**  il est aussi possible de renommer le fichier en le copiant (mon.txt) sur le disque local (toujours dans le répertoire courant) ... # **scp tux@192.168.1.103:/home/tux/rapport.texte ./monrapport.texte**  Pour copier un fichier d'un hôte vers un autre tout en étant sur une troisième station ... # **scp tux@srv01.profsavard.info:/home/tux/fichier.texte \ tux@srv02.profsavard.info:/home/tux/fichier.texte**  Dans le cas où le port SSH du serveur ne serait pas le port par défaut (22), il faut indiquer le port distant à utiliser ... # **scp -P <Port> home/tux/fichier.texte \ tux@srv02.profsavard.info:/home/tux/fichier.texte**  **Remarque** ... Lorsque l'on copie des fichiers ou des répertoires vers d'autres hôtes, il ne faut pas oublier que les fichiers ou répertoires deviendront propriété du compte avec lequel on se connecte à distance.  Pour préserver les propriétaire et groupe de chaque fichier ou répertoire, il sera donc utile de recourir à un logiciel tel que *tar* pour enregistrer l'intégralité des informations relatives à ce que l'on transfère.  **Gestion des clés**  Parfois les clefs des correspondants peuvent changer (réinstallation d’un hôte par exemple), on aura alors droit à ce message ...  ***@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@***  ***@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @***  ***@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@***  ***IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!  Someone could be eavesdropping on you right now (man-in-the-middle attack)!  It is also possible that the RSA host key has just been changed.  The fingerprint for the RSA key sent by the remote host is  xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx.  Please contact your system administrator.  Add correct host key in /home/<vous>/.ssh/known\_hosts to get rid of this message.  Offending key in /home/<vous>/.ssh/known\_hosts:4  RSA host key for <ip> has changed and you have requested strict checking.  Host key verification failed.***  Soit l'information est exacte et un hôte a été corrompu, ou bien il s'agit simplement d'un changement de clé (réinstallation du serveur par exemple) et dans ce cas il faut effacer les entrées dans le fichier ***.ssh/known\_hosts*** du compte.  Avant la chose était relativement simple, la clé était directement associée au nom ou à l'IP de l'hôte cible.  Ce n'est plus le cas à présent où elle est associée par UUID rendant quasiment impossible l'identification visuelle de la ligne concernée. Toutefois, ssh indique quelle est la ligne du fichier concernée.  Pour reprendre l'exemple précédent on peut lire la ligne  ***Offending key in /home/<Nom utilisateur>/.ssh/known\_hosts:4 →***  la clé en erreur est donc située ligne 4 du fichier ***.ssh/known\_hosts***.  Il existe cependant une méthode plus subtile en employant la commande suivante ... # **ssh-keygen -R <Adresse IP>** # **ssh-keygen -R srv01.profsavard.info** # **ssh-keygen -R 192.168.1.11 *# Host 192.168.1.11 found: line 1 /home/lsavard/.ssh/known\_hosts updated. Original contents retained as /home/lsavard/.ssh/known\_hosts.old***  Il est ainsi possible d'effacer seulement l'adresse IP concernée et relancer un ssh.  **Connexion à un répertoire */home* chiffré** Si on souhaite se connecter par SSH avec une clef publique sur un compte dont le répertoire personnel (*/home*) est chiffré, il est important de faire attention à ce que sur le serveur le fichier/répertoire .***ssh/authorized\_keys*** soit à la fois dans le */home* chiffré et déchiffré.  En effet si le fichier *authorized\_keys* est dans le répertoire personnel sous forme chiffré (*.private*), openSSH ne pourra pas lire la clef publique attendue.  Il faut donc créer un dossier *.ssh* et y mettre le fichier *authorized\_keys* quand le répertoire principal est démonté donc chiffré.  Cependant, si on ne le laisse pas aussi dans le répertoire personnel déchiffré et donc monté, la connexion SSH se fera avec la clef publique mais le répertoire personnel ne sera pas déchiffré automatiquement.  La meilleure solution est de créer des liens virtuels vers un dossier qui n'est pas soumis au chiffrement/déchiffrement.  **Authentification SSH avec plusieurs clés privées** Lorsque l'on se connecte à plusieurs serveurs, certains avec clés chiffrées, d'autres avec clé en clair, il faut pouvoir indiquer à SSH quelle clé on veut utiliser pour la connexion.  Pour indiquer au client SSH la clé qu'il doit utiliser pour chacun des serveurs, il faut créer le fichier ***~/.ssh/config*** (ou ***/etc/ssh/ssh\_config*** pour tous les utilisateurs du système) dans lequel il faut spécifier pour chacun des serveurs la clé qui doit être utilisée ... ***Host <Adresse du serveur sans phrase de reconnaissance> User <Nom utilisateur> IdentityFile ~/.ssh/<Clé sans phrase de reconnaissance>***  L'astuce pour différencier les clefs avec une phrase de reconnaissance (*passphrase*) de celles sans phrase de reconnaissance, serait de créer une clef de type ECDSA pour les identifications sans phrase de reconnaissance et DSA pour les identifications avec phrase de reconnaissance.  Ceci donnerait pour le premier ... ***IdentityFile ~/.ssh/id\_ecrsa.pub*** et pour le second ... ***IdentityFile ~/.ssh/id\_rsa.pub***  **Les empreintes (fingerprint)**  Retrouver l'empreinte de sa clef SSH, pour la communiquer à une personne qui veut se connecter et va utiliser sa clef publique ... # **ssh-keygen -l *Enter file in which the key is (/root/.ssh/id\_rsa): 2048 SHA256:3kuyyLoWBwMDxvXVFQ1hNA9I71p6GUphVi0GViui7kA root@srv01.profsavard.info (RSA)***  Ensuite la commande demande le fichier de la clef publique.  Sur un serveur on spécifiera ***/etc/ssh/ssh\_host\_rsa\_key.pub***.  **Configuration du serveur SSH**  La configuration par défaut du serveur SSH est suffisante pour fonctionner correctement.  Le fichier de configuration à éditer avec les droits d'administration est ***/etc/ssh/sshd\_config***.  Tableau des principales directives (à modifier le cas échéant)   |  |  |  |  | | --- | --- | --- | --- | | Directive du fichier | Valeur par défaut | Valeur possible | Effet de la valeur choisie | | Port | 22 | Tous les ports  non utilisés  de 1024 à 65535 | Permet d'éviter des désagréments avec les robots qui scannent Internet, notamment les ports par défaut | | PermitRootLogin | without-password prohibit-password | oui (yes) ou non (no) |  | | Protocol | 2 | 1 | Désigne la version du protocole ssh utilisé par le serveur. Par défaut, c'est la version 2 | | ListenAddress | 0.0.0.0 | Adresses IPv4 | Correspond à l'adresse locale d'écoute su serveur sshd en ipv4 | | ListenAddress | :: | Adresses IPv6 | Correspond à l'adresse locale d'écoute su serveur sshd en ipv6 | | PubkeyAuthentication | Oui (yes) | non (no) | Laisser oui (yes) si on veut établir l'authentification par clé | | PasswordAuthentication | Oui (yes) | Non (no) | On peut parfaitement conserver l'authentification par clé pour certains utilisateurs avec celle par mot de passe pour d'autres utilisateurs Conserver cette valeur à oui (yes) tant que l'authentification par clé n'est pas pleinement fonctionnelle, sinon on perdra toute connexion en SSH | | X11Forwarding | Oui (yes) | Non (no) | Laisser oui (yes) pour faire de  l'affichage graphique déporté | | #Banner /etc/issue.net | Ligne commentée donc inactive | Décommenter | Lorsque l'on essaie de se connecter à un serveur par SSH, le fichier /etc/issue.net est affiché (il est personnalisable comme salutations ou pour indiquer un avertissement, un guide, ...) | | #MaxStartups | 10:30:60 | Ligne commentée donc inactive | Décommenter (enlever symbole #) Le 10 représente le nombre de connexions acceptées sans qu'un utilisateur ait réussi à s'identifier, si cela passe au-dessus de 10, il y a 30 % de probabilités que les suivantes soient bloquées, et ce pourcentage augmente linéairement jusqu'à 100 % lorsque le maximum est atteint, à 60 connexions | | UsePAM | Oui (yes) | Non (no) | Mettre à no pour ne plus avoir à saisir un mot de passe avec l'usage des clés. Va de pair avec PubkeyAuthentication | | AllowUsers | Ligne absente (autorisé à tous) | Ajouter la ligne avec valeur(s) : AllowUsers Alice Bob | Spécifie les logins des seuls utilisateurs autorisés à se connecter. Idéal pour ouvrir un compte FTP à un uilisateur tout en restreignant l'accès au shell depuis SSH | | DenyUsers | Ligne absente (interdit à personne) | Ajouter la ligne avec valeur(s) | Interdit l'accès à SSH aux utilisateurs listés | | ClientAliveInterval | Ligne absente | Ajouter la ligne avec valeur en secondes : ClientAliveInterval 300 | Permet dans certains cas de maintenir une connexion sans coupures | | AllowGroups | Ligne absente (autorisé à tous les groupes) | Ajouter la ligne avec valeur(s) : AllowGroups groupname1 groupname2 | L'authentification via SSH ne sera possible que par des utilisateurs des groupes désigné par leur nom (pas par GID) | | DenyGroups | Ligne absente (interdit à aucun groupe) | Ajouter la ligne avec valeur(s) | Interdit l'accès à SSH aux utilisateurs des groupes listés | |  |  |  |  |   **En vrac ...**  **Autoriser/Interdire des utilisateurs** Il est possible d'autoriser une liste de certains utilisateurs à se connecter.  Il faut modifier ou ajouter cette ligne dans le fichier *sshd\_config* ... **AllowUsers utilisateur1 utilisateur2 utilisateur3 …**  Il est possible de refuser la connexion que de certains utilisateurs.  Il faut modifier ou ajouter cette ligne dans le fichier *sshd\_config* ... **DenyUsers utilisateur1 utilisateur2 utilisateur3 …**  **Infos pratiques côté Client** ~/.ssh/config : Configuration simple  Il existe un outil qui simplifie la gestion : les alias SSH. Si on a l'habitude de se connecter à plusieurs serveurs SSH, (noms d'hôtes différents, ports différents) et que l'on veut simplifier la saisie de commande, il est possible de créer des alias … # **ssh -p 5222 tux@rv01.profsavard.info**  Il suffit de créer le fichier ~/.ssh/config et d'y ajouter les lignes suivantes ... **Host srv01.profsavard.info  HostName 192.168.10.100  Port 5222  User tux**  Pour se connecter au serveur srv01.profsavard.info, il suffit de saisir la commande ... # **ssh srv01**  **Alias de tunnel** Il est même possible de créer un alias tunnel … **Host webmin**  **HostName 192.168.1.11**  **Port 22**  **User tuxexit**  **LocalForward 8080 192.168.1.10:10000**  Lors de la saisie de la commande *ssh webmin*, un tunnel SSH s'établit entre le client et le serveur ayant l'adresse IP 192.168.1.10. On a donc créé un tunnel SSH, qui va relier le port 8080 de le client sur le port 10000 de l'hôte distant.  Si le serveur n'a que le port 22 d'ouvert, c'est intéressant.  On peut accéder à *webmin* (qui écoute sur le port 10000) maintenant, avec le navigateur sur le client à l'adresse *https://127.0.0.1:8080*.  **Tunnels et redirection de ports**  **Redirection locale de port** L'option -L (*locale*) permet de rediriger un port distant vers une destination (hôte) locale.  La syntaxe de la commande est ... # **ssh <Hôte distant 01> -L <Port local>:<Hôte distant 02>:<Port distant>**  Exemple ... À partir d'un client qui a un accès en SSH sur un serveur (srv01-ssh) … Ce serveur est situé dans le même réseau qu'un serveur web (srv02-web avec l'adresse IP 192.168.1.50).  Ce serveur Web n'est pas atteignable directement depuis le réseau Internet.  On peut se connecter sur le srv01-ssh en effectuant une redirection de port 80 du srv02-web sur un port de l'hôte client ... # **ssh srv01ssh -L 1234:192.168.1.50:80**  Par la suite, il faut, sur l'hôte local, saisir sur l'hôte client dans un fureteur web ***http://127.0.0.1:1234***.  **Redirection distante de port** L'option -R distante (*remote*) permet de rediriger un port local vers une destination (hôte) distant. # **ssh -R <Port distant>:<Hôte local>:<Port distant> <Hôte distant>**  On a un serveur derrière un pare-feu (nommé srv01-ssh) et un serveur à la maison (srv03-maison) qui offre un accès SSH depuis Internet. Depuis srv01-ssh ... # **ssh -R 1234:localhost:22 srv03-maison** Depuis la maison, sur le serveur concerné ... # **ssh user-srv01-ssh@localhost -p1234**  Le mot de passe du serveur srv01-ssh sera demandé et on sera connecté. Cela s'avère très pratique pour ouvrir un tunnel inverse.  **Redirection dynamique de port (Proxy SOCKS)** L'option -D dynamique permet un transfert de port dynamique basé sur SOCKS.  # **ssh -D <port> <Serveur distant**>  Cette fonctionnalité est assez utile si on veut fureter à partir d'un proxy SOCKS.  On est connecté sur un point d'accès Wi-Fi public avec un périphérique.  Tout le flux est potentiellement non sécurisé et peut être modifié.  On peut alors, si on dispose d'un serveur SSH accessible (chez soi par exemple) se connecter à partir la commande ... # **ssh -D 8080 srv03-maison** Depuis un fureteur, on va dans les ***Options réseau***, et on sélectionne ***Configuration Manuelle du proxy*** et dans Hôte SOCKS on renseigne 127.0.0.1 et comme port 8080.  À partir de ce moment, le trafic web passe par le tunnel SSH (sécurisé depuis le périphérique client jusqu’à srv03-maison) puis sort par le lien réseau de srv03-maison.  **Taux-limite les connexions** Il est possible de limiter la vitesse à laquelle une adresse IP peut établir de nouvelles connexions SSH en configurant le pare-feu. Si une adresse IP tente de se connecter plus de 10 fois en 30 secondes, toutes les tentatives suivantes échoueront, car les connexions seront rejetées.  Dans un environnement mono-utilisateur, pour modifier le nombre total de connexions en connexion simultanées en attente (pas encore autorisées) au système, il faudra définir dans sshd\_config … **MaxStartups 2:30:10**  Cet exemple autorisera deux connexions en attente.  Entre la troisième et la dixième connexion, le système commence à supprimer de manière aléatoire les connexions de 30% à 100% lors de la dixième connexion simultanée.  Dans un environnement multi-utilisateur ou serveur, ces chiffres doivent être nettement plus élevés en fonction des ressources et de la demande en matière d'atténuation des attaques par déni d'accès. Définir un temps de connexion plus court (temps nécessaire pour conserver les connexions en attente en attente d'une autorisation) peut être une bonne idée car il libère les connexions en attente plus rapidement, mais au détriment de la commodité. **LoginGraceTime 30**  **Plus d'informations vers les journaux (*logs*)** Par défaut, le serveur openSSH se connecte à la fonction AUTH de *syslog*, au niveau INFO.  Si on souhaite enregistrer davantage d'informations, telles que les tentatives de connexion infructueuses, il faudra augmenter le niveau de journalisation à VERBOSE.  Il est recommandé de consigner plus d'informations si vous êtes intéressé par le trafic SSH malveillant. Pour augmenter le niveau, identifier la ligne suivante dans le ***sshd\_config*** … **LogLevel INFO** et modifier pour … **LogLevel VERBOSE**  Désormais, tous les détails des tentatives de connexion SSH seront enregistrés dans le fichier /var/log/auth.log.  Si on utilise un autre port que celi par défaut (TCP 22) ou si on pense que son serveur est suffisamment caché pour ne pas à avoir besoin de beaucoup de sécurité, il est possible d’augmenter son niveau de journalisation et examiner son fichier auth.log de temps à autre.  Si on rencontre un nombre important de tentatives de connexion frauduleuses, l’hôte est attaqué et on aura besoin de plus de sécurité.  Quelles que soient les précautions de sécurité qui ont été prises, il est possible de définir le niveau d’enregistrement sur VERBOSE pour une semaine et connaître le trafic parasite généré. Ainsi, il sera possible de constater à quel point l’hôte est attaqué.  **Affichage d’une bannière** Il peut être intéressant, du point de vue juridique, d’afficher un message de mise en garde lors de la connexion à un hôte depuis SSH.  Pour ajouter une bannière qui sera affichée avant l'authentification, identifier cette ligne … **#Banner /etc/issue.net** et le remplacer par ,,, **Bannière /etc/issue.net**  Cela affichera le contenu du fichier ***/etc/issue.net***, qu’il est possible d’éditer selon ses besoins.  Si on souhaite afficher la même bannière pour les utilisateurs SSH et pour les utilisateurs se connectant à une console locale … Banner /etc/issue  Exemple de bannière …  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **ALERT!  You are entering into a secured area!  Your IP, Login Time, Username has been noted and has been sent to the server administrator. This service is restricted to authorized users only. All activities on this system are logged. Unauthorized access will be fully investigated and reported to the appropriate law enforcement agencies.**  **ALERTE!  Vous entrez dans une zone sécurisée!  Votre adresse IP, heure de connexion, nom d'utilisateur a été noté et a été envoyé à l'administrateur du serveur. Ce service est réservé aux utilisateurs autorisés. Toutes les activités sur ce système sont enregistrées. Les accès non autorisés feront l’objet d’une enquête approfondie et seront signalés aux services de détection et de répression appropriés.**  **¡ALERTA!  Usted está entrando en un área segura!  Su IP, tiempo de inicio de sesión, nombre de usuario se anotó y se envió al administrador del servidor. Este servicio está restringido a usuarios autorizados solamente.  Todas las actividades en este sistema están registradas. El acceso no autorizado se investigará por completo y se informará a las agencias de aplicación de la ley correspondientes.**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **Annexe 1 RSA, DSA ou ECDSA**  Il existe de nombreux algorithmes de chiffrement basés sur des clés asymétriques, mais les deux dominants, et de loin, son RSA et DSA, d’après les initiales de leurs auteurs respectifs. RSA est historiquement le premier, et DSA a été développé plus récemment, comme une alternative alors qu'RSA était encore breveté. ECDSA (*Elliptic Curve Digital* *Signature Algorithm*), quant à lui, est un algorithme de signature numérique à clé publique, variante de DSA. Il fait appel au chiffrement à l’aide les courbes elliptiques.  RSA demeure le plus populaire, mais on peut choisir librement RSA, DSA ou ECDSA pour ses clés, du moment que l’on utilise une taille suffisante.  On n’aura habituellement pas besoin des trois et il est possible les supprimer, mais un client SSH ne pourra vérifier l'empreinte d’un serveur que si une empreinte prise en charge est enregistrée dans le fichier *~/.ssh/known\_hosts*.  Au minimum, les clés RSA doivent être conservées.  Tous les clients ne prennent pas en charge ECDSA car il présente des problèmes de sécurité et n'est généralement plus recommandé. Il n'y a aucun inconvénient à les garder tous, cependant.   * RSA est bien considéré et soutenu partout. Il est considéré comme assez sécurisé.  Les tailles de clé communes vont jusqu'à 4096 bits et aussi basses que 1024.  La taille de la clé est ajustable.  On devrait privilégier RSA. * DSA n'est plus d'usage courant, car un mauvais caractère aléatoire lors de la génération d'une signature peut entraîner la fuite de la clé privée.  Dans le passé, il était garanti qu'il fonctionnerait partout conformément à la RFC 4251, mais ce n'est plus le cas. DSA a été normalisé à seulement 1024 bits (dans FIPS 186-2, bien que FIPS 186-3 ait augmenté cette limite). openSSH 7.0 et versions ultérieures désactivent cet algorithme. * ECDSA est plus récent et est basé sur DSA.  Il présente les mêmes faiblesses que DSA, mais il est généralement considéré comme plus sécurisé, même pour des clés plus petites.  Il utilise les courbes NIST (P256). * Ed25519, bien que non répertorié, est disponible sur les installations openSSH les plus récentes.  Il est similaire à ECDSA mais utilise une courbe supérieure et ne présente pas les mêmes faiblesses lorsque des RNG faibles sont utilisés comme DSA/ECDSA.  Il est généralement considéré comme mathématiquement le plus fort.   **Annexe 2 Paramètres de sshd\_conf**  Précisions at autres paramètres   * ***AllowUsers*** Ce mot-clef peut être suivi d'une liste de motifs de noms d'utilisateurs, séparés par des espaces.  S'il est spécifié, seuls les noms d'utilisateurs correspondant à un des motifs sont autorisés à se connecter.  On peut utiliser les caractères « \* » ou « ? » comme des jokers.  Seuls les noms d'utilisateurs sont valides ; les identifiants d'utilisateurs (UID) ne sont pas reconnus.  Par défaut, la connexion est autorisée pour tous les utilisateurs.  Si le motif est de la forme UTILISATEUR@HÔTE, alors UTILISATEUR et HÔTE sont vérifiés séparément, en restreignant les connexions à des utilisateurs en particulier provenant d'hôte en particulier. * ***AllowGroups*** Ce mot-clef peut être suivi d'une liste de motifs de noms de groupes, séparés par des espaces.  S'il est spécifié, seuls les utilisateurs dont le groupe principal ou les groupes supplémentaires correspondent à un des motifs sont autorisés à se connecter.  On peut utiliser les caractères « \* » ou « ? » comme jokers.  Seuls les noms de groupes sont valides ; les identifiants de groupes (GID) numériques ne sont pas reconnus.  Par défaut, la connexion est autorisée pour tous les groupes. * ***DenyUsers*** Ce mot-clef est suivi d'une liste de motifs de noms d'utilisateurs, séparés par des espaces.  Les utilisateurs dont le nom correspond à un des motifs ne sont pas autorisés à se connecter.  Dans les motifs, on peut utiliser les caractères « \* » et « ? » comme des jokers.  On ne spécifie que des noms d'utilisateurs ; les identifiants numériques d'utilisateurs ne sont pas autorisés.  Par défaut, tous les utilisateurs sont autorisés à se connecter. Si le motif est de la forme UTILISATEUR@HÔTE, UTILISATEUR et HÔTE sont vérifiés séparément, et la connexion est restreinte à certains utilisateurs se connectant de certaines machines. * ***DenyGroups*** Ce mot-clef est suivi d'une liste de motifs de noms de groupes, séparés par des espaces.  Les utilisateurs dont le groupe principal ou les groupes secondaires correspondent à un des motifs ne sont pas autorisés à se connecter.  Dans les motifs, on peut utiliser les caractères « \* » et « ? » comme des jokers.  On ne spécifie que des noms de groupes ; les identifiants numériques de groupes ne sont pas autorisés. Par défaut, tous les groupes sont autorisés à se connecter. * ***HostKey*** Définit le chemin de la clé privée du serveur. Par défaut, /etc/ssh/ssh\_host\_rsa\_key. * ***UsePrivilegeSeparation*** Spécifie si sshd sépare les privilèges en créant un processus fils non privilégié pour prendre en charge le trafic réseau entrant.  Après une authentification réussie, un autre processus est créé avec les privilèges de l'utilisateur authentifié.  Le but de la séparation de privilèges est d'éviter l'escalade de privilèges si le processus non privilégié est corrompu.  Par défaut, Oui (yes). * ***KeyRegenerationInterval*** Dans la version 1 du protocole, la clef éphémère du serveur est regénérée automatiquement après ce nombre de secondes (si elle a été utilisée).  Le but de la régénération est d'éviter le déchuffrement de sessions capturées en s'introduisant plus tard sur la machine et en volant la clef.  La clef n'est jamais stockée nulle part.  Si la valeur est 0, la clef n'est jamais regénérée.  Par défaut 3600, (secondes). * ***ServerKeyBits*** Définit le nombre de bits de la clef éphémère pour la version 1 du protocole.  La valeur minimale est 512 et la valeur par défaut est 768. * ***SyslogFacility*** Donne le code de facilité utilisé lors de l'enregistrement des messages du démon sshd Les valeurs possibles sont : DAEMON, USER, AUTH, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7.  Par défaut, AUTH. * ***LogLevel*** Donne le niveau de verbosité utilisé lors de l'enregistrement des messages du démon sshd.  Les valeurs possibles sont : QUIET, FATAL, ERROR, INFO, VERBOSE, DEBUG, DEBUG1, DEBUG2 and DEBUG3.  Par défaut INFO. DEBUG et DEBUG1 sont équivalents.  DEBUG2 et DEBUG3 spécifient des niveaux plus élevés de sortie de débogage.  L'enregistrement à l'aide d'un niveau DEBUG a tendance à empiéter sur la vie privée des utilisateurs et n'est pas recommandé. * ***LoginGraceTime*** Désigne la durée d'inactivité (en minutes) au bout de laquelle le serveur se déconnecte automatiquement.  Si la valeur est 0 (zéro), le serveur ne se déconnecte jamais.  Par défaut, 120. * ***PermitRootLogin*** Définit si oui ou non le super-utilisateur root possède l'autorisation de se connecter par ssh. Les arguments sont : ***yes*** | ***without-password*** | ***forced-commands-only*** | ***no***.  Par défaut, yes. Si cette option est réglée à ***without-password***, l'authentification par mot de passe est désactivée pour *root*.  Si cette option est réglée à ***forced-commands-only***, les connexions de *root* sont autorisées avec une authentification par clef publique, mais seulement si l'option *command* est spécifiée (ce qui peut être utile pour effectuer des sauvegardes à distance même si les connexions de root sont normalement interdites). Toutes les autres méthodes d'authentification sont désactivées pour *root*.  Si cette option est réglée à Non (no), root n'est pas autorisé à se connecter. * ***StrictModes*** Spécifie si sshd doit vérifier les modes et le propriétaire des fichiers de l'utilisateur et du répertoire de base (*home directory*) de l'utilisateur avant d'accepter une connexion.  C'est normalement souhaitable, parce que quelque fois, les novices laissent accidentellement leur répertoire ou leurs fichiers en accès complet à tout le monde.  Par défaut, Oui (yes). * ***RSAAuthentication*** Spécifie si on autorise la pure authentification RSA. Par défaut Oui (yes). Cette option ne s'applique qu'à la version 1 du protocole. * ***PubkeyAuthentication*** Spécifie si on autorise l'authentification par clef publique, par opposition à l'authentification par mot de passe. Par défaut, Oui (yes).  Cette option ne s'applique qu'à la version 2 du protocole. * ***AuthorizedKeysFile*** Indique le chemin du fichier contenant les clés publiques des utilisateurs distants. * ***IgnoreRhosts RhostsRSAAuthentication*** Spécifie que l'on n'utilise pas les fichiers *.rhosts* et *.shosts* pour les authentification activées par les options *RhostsAuthentication* et *RhostsRSAAuthentication* ou *HostbasedAuthentication*.  Les fichiers */etc/hosts.equiv* et */etc/ssh/shosts.equiv* sont néanmoins utilisés.  Par défaut, Oui (yes). * ***HostbasedAuthentication*** Spécifie si on autorise une authentification par *rhosts* ou */etc/hosts.equiv* conjointement avec une authentification de machine cliente réussie par clef publique (authentification par hôte).  Cette option est similaire à l'option *RhostsRSAAuthentication* et ne s'applique qu'à la version 2 du protocole. Par défaut, Non (no). * ***IgnoreUserKnownHosts*** Spécifie si sshd doit ignorer le fichier *$HOME/.ssh/known\_hosts* de l'utilisateur lors des authentifications des options *RhostsRSAAuthentication* ou *HostbasedAuthentication*.  Par défaut, Non (no). * ***PermitEmptyPasswords*** Définit si le serveur accepte la connexion à un compte utilisateur ne possédant pas de mot de passe.  Par défaut, Non (no). * ***ChallengeResponseAuthentication*** Spécifie si on autorise l'authentification par stimulation-réponse (*challenge response*).  Toutes les formes d'authentification de *login.conf* sont gérées.  Par défaut, Oui (yes). * ***PasswordAuthentication*** Permet d'autoriser l'authentification par mot de passe.  Par défaut, Oui (yes). * ***X11Forwarding*** Permet de rediriger les sorties X11.  Cela permet donc, par exemple, d'ouvrir des applications graphiques à distance.  La redirection X11 est automatiquement désactivée si l'option *UseLogin* est activée.  Par défaut, Non (no). * ***X11DisplayOffset*** Spécifie le premier numéro d'affichage disponible pour les redirections X11 de sshd.  Ceci évite à sshd d'interférer avec les vrais serveurs X11.  Par défaut, 10. * ***PrintMotd*** Permet d'afficher le contenu du fichier */etc/motd (Message of the day* - Message du jour) à la connexion.  Par défaut, oui (yes). * ***PrintLastLog*** Permet d'afficher la date et l'heure de la dernière connexion.  Par défaut, Oui (yes).à * ***TCPKeepAlive*** Permet de garder ouverte une connexion ssh existante grâce à l'envoi d'un paquet chiffré par ssh.  Par défaut, Oui (yes). * ***UseLogin*** Spécifie si on utilise *login* pour les connexions à des sessions interactives.  Par défaut, Non (no).  Notes … On n'utilise jamais *login* pour l'exécution de commandes à distance.  Si cette option est activée, on désactive X11Forwarding parce que *login* ne sait pas traiter les *cookies* *xauth*.  Si on spécifie l'option UsePrivilegeSeparation, elle sera désactivée après l'authentification. * ***MaxStartups*** Spécifie un nombre maximal de connexions concurrentes au démon sshd non authentifiées.  Les connexions supplémentaires sont purgées si elles ne peuvent pas s'authentifier ou si le délai de grâce défini à l'aide de l'option *LoginGraceTime* expire pour une connexion.  Par défaut 10.  Par ailleurs, on peut activer une purge hâtive aléatoire en spécifiant un triplet *début:taux:total*  (par exemple, 10:30:60).  sshd refuse les tentatives de connexion avec une probabilité de taux/100 » (30 %) s'il y a « début (10) connexions non authentifiées en cours.  La probabilité augmente linéairement et toutes les tentatives de connexion sont refusées si le nombre de connexions non authentifiées atteint « total » (60). * ***Banner*** Pour certaines juridictions, l'envoi d'un message avant l'authentification est nécessaire pour disposer d'une protection légale. Le contenu du fichier spécifié est envoyé à l'utilisateur distant avant d'autoriser la connexion. Cette option n'est disponible qu'avec la version 2 du protocole. Par défaut, on n'affiche pas de message. * ***AcceptEnv*** Spécifie quelles variables d'environnement envoyées par le client seront copiées dans l'environnement de la session.  Cette option n'est supportée que par la version 2. * ***Subsystem sftp*** Configure un sous-système externe (par exemple un démon de transfert de fichiers).  Les arguments doivent être un nom de sous-système et une commande à exécuter lors d'une requête à ce sous-système.  La commande sftp-server implémente le sous-système de transfert de fichiers sftp.  Par défaut, aucun sous-système n'est défini.  Note … Cette option ne s'applique qu'à la version 2 du protocole.   **Annexe 3 Authentification sans mot de passe**  L'authentification sans mot de passe est très pratique et pratiquement incontournable pour exécuter des scripts distants. L'authentification sans mot de passe est donc très efficace mais pose cependant un problème de sécurité non négligeable.  Son principe est tout simple … Cela consiste en la génération d'un couple de clé privée/clé publique, de la même manière que pour l'authentification par échange de clés et en la diffusion de votre clé publique.  Toutefois, au moment de saisir une phrase de reconnaissance (*passphrase*) comme il l'est demandé au cours de la génération du couple de clés, il suffit de ne pas saisir de mot de passe (seulement la touche Entrée) lorsque cela est demandé par openSSH. Il suffit de répéter cette touche à la demande de re-saisie de la phrase de reconnaissance.  Cela a pour effet de donner une chaîne vide comme phrase de reconnaissance, ce qui revient à dire qu'il n'y a pas de phrase de reconnaisasance. Ainsi, lors de la connexion par ssh, l'étape de demande de phrase de reconnaissance est omise et la connexion s'effectue sans demande de vérification au clavier.  **Annexe 4 Automatiser l'utilisation de *ssh-agent***  L’agent SSH (*ssh-agent*) n'est valable que pour la console dans laquelle il est lancé. Ainsi, pour optimiser l'utilisation de *ssh-agent* et éviter de devoir recommencer la procédure de lancement de *ssh‑agent* et la copie des variables d'environnement à chaque fois, on peut écrire un bout de code pour pallier ce problème.  Ce code est à coller dans le fichier *.bash\_profile* de son répertoire personnel (exemple : ***/home/tux/.bash\_profile***.  **SSH\_ENV="$HOME/.ssh/environment"**  **function start\_agent {  echo "Initialising new SSH agent..."  /usr/bin/ssh-agent | sed 's/^echo/#echo/' > "${SSH\_ENV}"  echo succeeded  chmod 600 "${SSH\_ENV}"  . "${SSH\_ENV}" > /dev/null  /usr/bin/ssh-add; }**  **# Source SSH settings, if applicable if [ -f "${SSH\_ENV}" ]; then  . "${SSH\_ENV}" > /dev/null  ps -ef | grep ${SSH\_AGENT\_PID} | grep ssh-agent$ > /dev/null || {  start\_agent;  } else  start\_agent; fi**  Ainsi, avec ce code, il n'est plus nécessaire de lancer ssh-agent. Il suffit de copier sa clé sur le serveur si ce n'est pas déjà fait et se connecter en ssh sans besoin de mot de passe.  Astuce … On peut consulter un code amélioré (de l’auteur Joseph M. Reagle) à l’adresse suivante …  <https://github.com/robbyrussell/oh-my-zsh/blob/master/plugins/ssh-agent/ssh-agent.plugin.zsh>  **Annexe 5 Serveurs de clés publiques**  **Publication de sa clef publique** Les clés asymétriques ne sont pas seulement utiles pour SSH et les systèmes s’y greffant. Elles sont aussi, par exemple, le moyen privilégié pour (dé)chiffrer ou signer numériquement (ou les deux) les courriels.  Il est donc utile de pouvoir facilement récupérer la clé publique de quelqu’un …   * soit parce qu’on veut lui envoyer un contenu confidentiel (on va donc le chiffrer avec la clé publique du destinataire, dont seule la clé privée pourra déchiffrer) ; * soit parce qu’on a reçu un courriel signé numériquement de cette personne (grâce à sa clé privée), et qu’on veut vérifier l’exactitude de l’envoyeur, et que le contenu n’a pas été altéré (ce qu’on peut faire uniquement à l’aide de sa clé publique).   Bien entendu, la personne en question peut avoir, au préalable, transmis sa clé publique. Encore faut-il pouvoir faire confiance au mécanisme qui a fait cette transmission. Un autre moyen plus général est, pour la personne, de publier sa clé publique aux yeux de tous.  Pour cela, il existe des serveurs de clés. La plupart sont gratuits, et les plus connus sont répertoriés par les principaux systèmes exploitant des clés. En voici quelques-uns :  Serveurs disponibles sur la toile …   * keyserver.pgp.com  <https://keyserver.pgp.com/vkd/GetWelcomeScreen.event> * pgp.webtru.st/  <https://peegeepee.com/?utm_source=co-domain&utm_medium=legacy&utm_campaign=pgp.webtru.st> * pgpkeys.mit.edu  <http://pgpkeys.mit.edu/>     *Serveur de distribution de clés keyserver.pgp.com*  **Annexe 6 Fichiers de configuration d'openSSH**  openSSH est constitué de deux ensembles de fichiers de configuration, comme c'est en général le cas sur les services sous linux.  Il y a un fichier de configuration pour les programmes clients (*ssh*, scp et sftp) et l'autre pour le service serveur(*sshd*). Les informations de configuration SSH qui s'appliquent à l'ensemble du système sont emmagasinés dans le répertoire */etc/ssh*.  Voici les principaux fichiers de configuration …   * ***ssh\_config*** Fichier de configuration client SSH pour l'ensemble du système. Il est écrasé si un même fichier est présent dans le répertoire personnel de l'utilisateur (*~/.ssh/config*) ; * ***sshd\_config*** Fichier de configuration pour *sshd* ; * ***ssh\_host\_dsa\_key*** Clé DSA privée utilisée par sshd ; * ***ssh\_host\_dsa\_key.pub*** Clé DSA publique utilisée par sshd ; * ***ssh\_host\_rsa\_key*** Clé RSA privée utilisée par sshd pour la version 2 du protocole SSH ; * ***ssh\_host\_rsa\_key.pub*** Clé RSA publique utilisée par sshd pour la version 2 du protocole SSH ; * ***ssh\_host\_key*** Clé RSA privée utilisée par sshd pour la version 1 du protocole SSH ; * ***ssh\_host\_key.pub*** Clé RSA publique utilisée par sshd pour la version 1 du protocole SSH.   Les informations spécifiques à un utilisateur sont stockées dans son répertoire personnel à l'intérieur du répertoire ***~/.ssh/*** …   * ***authorized\_keys*** ou parfois ***authorized\_keys2*** Ce fichier contient une liste de clés publiques autorisées Si un utilisateur se connecte et prouve qu'il connaît la clé privée correspondant à l'une de ces clés, il obtient l'authentification. Notez qu'il ne s'agit que d'une méthode d'authentification facultative ; * ***id\_dsa*** Contient l'identité d'authentification DSA de l'utilisateur ; * ***id\_dsa.pub*** Clé DSA publique de l'utilisateur ; * ***id\_rsa*** Clé RSA publique utilisée par sshd pour la version 2 du protocole SSH ; * ***identity*** Clé RSA privée utilisée par sshd pour la version 1 du protocole SSH ; * ***known\_hosts*** Ce fichier contient les clés hôte DSA des serveurs SSH auxquels l'utilisateur s'est connecté.   **Annexe 7 Reverse SSH**  Il est simple de configurer un serveur SSH et de s'y connecter lorsque le pare-feu/routeur est correctement configuré.  Cependant certains réseaux ne laissent pas par défaut entrer les connexions SSH, le port par défaut (22) n'étant pas ouvert. Les connexions sortantes ne sont toutefois pas bloquées, ce qui va permettre de se connecter quand même.  Une parade est donc d'utiliser la fonction REVERSE de SSH. Le principe consiste à initier une connexion depuis la machine derrière le routeur sur un hôte tierce, et ainsi permettre une connexion retour depuis l’hôte tierce qui ne sera pas bloquée.  Cette façon de procéder est très utile pour dépanner quelqu'un à distance qui aura juste à initier la connexion sortante en entrant une ligne depuis le terminal, sans avoir à configurer le pare-feu/routeur. Il n'est également pas nécessaire de connaitre l'adresse IP de la machine distante ni d'effectuer un routage de la connexion.  **Connexion directe**  À partir del'exemple de configuration suivant …  userD@distant est le serveur SSH à joindre mais est inaccessible dû aux paramétrages du routeur/pare-feu ...  userD@distant est le serveur SSH Ã  joindre mais est inaccessible dÃ» aux paramÃ©trages du routeur/pare-feu...   * userD@distant correspond à l'utilisateur userD, sur le poste appelé distant qui a les ports entrants bloqués et donc inaccessible depuis l'extérieur. * userL@local correspond à l'utilisateur userL, sur le poste appelé local qui dispose également d'un serveur SSH, et qui va permettre d'accéder à la machine distant   Le principe de connexion à SSH est habituellement basé sur le système du Client local qui se connecte au Serveur distant.  **Note** … Ne pas oublier qu'ici c'est le Serveur distant qui se connecte au Client local.  **Création d'un utilisateur dédié sur le poste local** Comme première tâche, il faut créer un nouvel utilisateur spécialement pour cette connexion afin que l'utilisateur userD du poste distant ne puisse pas avoir un accès complet au poste local.  Ce nouvel utilisateur créé pourra cependant avoir des droits personnalisés.  # **adduser --no-create-home tuxlocal** où … ***--no-create-home*** est l'option spécifiée pour ne pas créer de dossier */home/userL* sur le poste local.  *userL* est à remplacer par le nom de son choix mais suffisamment explicite pour savoir sur quel hôte est l’hôte .  Le mot de passe créé servira pour se connecter lors de l'étape suivante.  **Connexion sur le poste local depuis le poste distant** À cette étape, il faut Initier une connexion sur local … # **ssh -R 12345:localhost:22 tuxlocal@local** où 12345 est à remplacer par un numéro de port aléatoire (entre 1024 et 65535 qui sont réservés pour des applications utilisateurs) et non utilisé de son choix et tuxlocal et le mot de passe de connexion sont ceux défini précédemment.  **Connexion sur le poste distant depuis le poste local** La connexion étant désormais activée depuis distant sur local, le pare-feu va donc laisser rentrer la connexion reverse, à savoir depuis local sur distant.  # **ssh -p 12345 tuxlocal@localhost** où 12345 est le port choisi auparavant et userD est à remplacer par le nom d'utilisateur permettant de se connecter au serveur ssh sur distant.  **Astuces** … Afin de faciliter la connexion, il est suggéré de créer le même compte (ici tuxlocal), afin de faciliter la gestion des mots de passe.  **Connexion par un serveur tiers** Cette configuration est pratique quand le poste local est lui-même derrière un pare-feu et/ou ne dispose pas d'un serveur SSH.  userD@distant et userL@local ne sont pas accessibles depuis l'extÃ©rieur  userD@distant et userL@local ne sont pas accessibles depuis l'extérieur   * ***userD@distant*** correspond à l'utilisateur userD, sur le poste appelé distant qui a les ports entrants bloqués et donc inaccessible depuis l'extérieur * ***userS@serveur*** correspond à l'utilisateur userS, sur le poste appelé serveur qui dispose d'un accès libre à son serveur ssh * ***userL@local*** correspond à l'utilisateur userL, sur le poste appelé local qui va accéder à la machine serveur pour atteindre distant   Pour résumer le principe, il s'agira de …   * connecter distant sur serveur ; * connecter local sur serveur ; * depuis le terminal qui a initié la connexion local sur serveur pour atteindre distant.   **Création d'un utilisateur dédié sur le poste serveur** Cette partie est facultative si l’hôte serveur dispose déjà d'un utilisateur public.  # **adduser --no-create-home userS** où … l'option ***--no-create-home*** est spécifiée pour ne pas créer de dossier */home/userS* sur le poste serveur.  **Connexion sur le poste serveur depuis le poste distant** Il faut maintenant Initier une connexion sur le serveur … # **ssh -R 12345:localhost:22 userS@serveur** où … 12345 est à remplacer par un numéro de port aléatoire de votre choix, le port 22 est le port d'écoute ssh sur la machine distant, userS et le mot de passe de connexion sont ceux défini précédemment, serveur est l'adresse IP ou le nom de domaine du serveur tiers.  **Astuce** … L'option -N peut également être ajoutée pour ne pas faire apparaitre d'invite de terminal sur distant.  Connexion sur le poste serveur depuis le poste local  **Création du pont entre serveur et local # ssh userS@serveur** où … userS et le mot de passe de connexion sont ceux défini précédemment, serveur est l'adresse ip ou le nom de domaine du serveur tiers,  **Accès à la machine distante depuis la machine locale** Il est désormais possible d’atteindre le poste distant en saisissant dans le terminal du poste local connecté précédemment sur serveur … # **ssh -p 12345 userD@localhost**  Références  <https://doc.ubuntu-fr.org/ssh> <http://www.linux-france.org/prj/edu/archinet/systeme/index.html> <https://doc.ubuntu-fr.org/tutoriel/reverse_ssh> <https://fr.wikipedia.org/wiki/OpenSSH> <https://www.openssh.com/manual.html>   |  |  | | --- | --- | |  | **https://creativecommons.org/licenses/by-sa/4.0/** | |
|  |  |  |