|  |  |
| --- | --- |
| SSH Client, SSH Terminal, SSH Android, SSH Client APK SSH Download for  Android – Download SSH Client, SSH Terminal, SSH Android, SSH Client APK  Latest Version - APKFab.com | Secure Shell |

Table des matières

[Serveur SSH 3](#_Toc90808417)

[Introduction 3](#_Toc90808418)

[openSSH 4](#_Toc90808419)

[Établissement d'une connexion SSH 5](#_Toc90808420)

[Première connexion 5](#_Toc90808421)

[Contraintes SSH 6](#_Toc90808422)

[Installation et configuration de base SSH 6](#_Toc90808423)

[Installation du serveur SSH 6](#_Toc90808424)

[Fichiers openSSH 6](#_Toc90808425)

[Utilisation du serveur SSH 6](#_Toc90808426)

[Première connexion avec SSH Authentification par mot de passe 7](#_Toc90808427)

[Configuration du serveur SSH 8](#_Toc90808428)

[Directives qui affectent l'accès utilisateur 9](#_Toc90808429)

[Utilisation d’openSSH 11](#_Toc90808430)

[Fichier ssh\_config 11](#_Toc90808431)

[Commande ssh 12](#_Toc90808432)

[Connexion à un serveur SSH 12](#_Toc90808433)

[Commandes scp et sftp 13](#_Toc90808434)

[Fonctionnalités SSH avancées 14](#_Toc90808435)

[Connexion sans mot de passe 14](#_Toc90808436)

[Suppression des clés autorisées du serveur 15](#_Toc90808437)

[Gestion des clés à l’aide de Putty 16](#_Toc90808438)

[Création une clé SSH à l’aide de PuTTY (sous Windows) 16](#_Toc90808439)

[Utilisation de la clé publique avec Putty 17](#_Toc90808440)

[Suppression des clés autorisées du serveur 18](#_Toc90808441)

[En vrac 19](#_Toc90808442)

[Taux-limite les connexions 19](#_Toc90808443)

[Plus d'informations vers les journaux (*logs*) 19](#_Toc90808444)

[Affichage d’une bannière 19](#_Toc90808445)

[Paramètres de sshd\_conf 21](#_Toc90808446)

# Serveur SSH

## 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 donc impossible d'utiliser un renifleur comme Wireshark pour voir ce que fait l'utilisateur.

Le protocole SSH a été conçu avec l'**objectif de remplacer les différents programmes** **rlogin**, **telnet**, **rcp**, **ftp** et **rsh**.

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 hôte afin d'obtenir un interpréteur de commandes (*shell*).   
  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.

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 principaux usages de base de SSH sont ...

* l'**accès** **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 ;
* le **déport de l'affichage graphique** de l’hôte distant ;
* le **transfert des fichiers** en ligne de commande ;
* le **montage ponctuel d’un répertoire distant**, soit en ligne de commande, soit à l’aide d’un interface graphique ;
* le **montage automatique de répertoires distants**.

## openSSH

Le **paquet le plus répandu** utilisé sous Linux pour fournir des connexions rapides et sécurisées aux serveurs Linux est **openSSH**.   
Le paquet openSSH utilise le protocole Secure Shell (SSH) pour **établir une connexion chiffrée entre deux périphériques réseau**.

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 **répertoire /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.

La suite logicielle OpenSSH inclut les outils suivants ..

* **ssh**  
  Remplaçant pour les clients rlogin et telnet ...  
  **>> ssh <utilisateur>@<Adrresse IP ou Nom DNS du serveur>**
* **scp**  
  Remplaçant pour le client rcp ...  
  **>>** **scp** **<utilisateur>@<Adrresse IP ou Nom DNS du serveur>:<Nom de fichier>**
* **sftp**  
  Remplaçant pour le client ftp ...  
  **>>** **sftp <utilisateur>@<Adrresse IP ou Nom DNS du serveur>**
* **sshd**  
  Démon (*daemon*) SSH
* **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.
* **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.

## Établissement d'une connexion SSH

Un serveur SSH dispose d'un couple de clefs RSA stocké dans le répertoire /etc/ssh/ et généré lors de l'installation du serveur.

* Le **fichier ssh\_host\_rsa\_key** contient la clef privée et possède les droits 600 ;
* Le **fichier ssh\_host\_rsa\_key.pub** contient la clef publique et possède les droits 644.

Les étapes de l'établissement d'une connexion SSH sont …

1. Le **serveur** **envoie sa clef publique au client**.   
   Celui-ci **vérifie qu'il s'agit bien de la clef du serveur**, s'il l'a déjà reçue lors d'une connexion précédente ;
2. Le **client génère une clef secrète** (chiffrement symétrique) et l'envoie au serveur, en chiffrant cet échange avec la clef publique du serveur (chiffrement asymétrique).   
   Le **serveur déchiffre cette clef secrète en utilisant sa clé privée**, ce qui confirme qu'il est bien le vrai serveur ;
3. Afin de le prouver au client, il **chiffre un message standard avec la clef secrète et l'envoie au client**.  
   Si le client **retrouve le message standard en utilisant la clef secrète**, il a la preuve que le **serveur est bien le vrai serveur** ;
4. Une fois la **clef secrète échangée**, le **client et le serveur peuvent alors établir un canal sécurisé** grâce à la clef secrète commune (chiffrement symétrique) ;
5. Une fois que le **canal sécurisé est en place**, le client va pouvoir envoyer au serveur une demande d’ouverture de session et le mot de passe de l'utilisateur pour vérification.   
   Le **canal sécurisé reste en place jusqu'à ce que l'utilisateur se déconnecte**.

## Première connexion

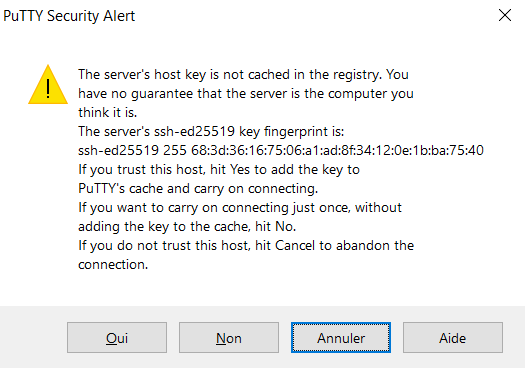
Si c'est la **première connexion SSH** depuis ce client vers ce serveur, **ce dernier demande si l’empreinte numérique** (*fingerprint*) de la clef publique **présentée par le serveur est bien la bonne**.   
Afin de **s’assurer que l’on se connecte au bon serveur**, on doit connaître de façon certaine l’empreinte numérique de sa clef publique et la comparer à celle qui s’affiche.   
Si les deux empreintes numériques sont identiques, on **confirme la connexion par yes** et la clef publique du serveur est alors **rajoutée au fichier local ~/.ssh/known\_hosts**.  
Si ce n’est pas la première connexion depuis ce client vers le serveur, puisque sa clef publique est déjà dans le fichier **~/.ssh/known\_hosts**, il n’est rien demandé.

Ensuite, on **saisit le mot de passe de l’utilisateur** et on voit **apparaître l’invite de commandes**, comme si on s’était connecté en local sur l‘hôte distant.

La seule **contrainte est de s'assurer que la clef publique présentée par le serveur est bien sa clef publique**… sinon le client risque de se connecter à un faux serveur qui aurait pris l'adresse IP du vrai serveur (ou toute autre magouille).   
Une bonne méthode est de **demander à l'administrateur du serveur quelle est l’empreinte numérique** (*fingerprint*) de la **clef** **publique du serveur** avant de s'y connecter pour la première fois.   
Cette **empreinte numérique d'une clef publique est une chaîne de 32 caractères hexadécimaux** à peu près unique pour chaque clef (un hachage) ;

On peut obtenir cette empreinte numérique à l’aide de la commande …  
**>> sudo ssh-keygen -lf /etc/ssh/ssh\_host\_rsa\_key.pub  
2048 SHA256:eq+ckurqbjfAhttuYeRqMxHuYVdkr/DmcYE9hnT5afg root@vps-0d518624 (RSA)**

Si vous utilisez Putty ce message apparaît …



## Contraintes SSH

Il y a **trois contraintes majeures** pour garder un système sécurisé après avoir installé un serveur SSH …

* **mettre à jour un serveur SSH à jour au niveau de la sécurité**, ce qui doit être le cas si vous faites consciencieusement les mises à jour de sécurité en suivant la procédure ;
* **s’assurer que les mots de passe** de **TOUS les utilisateurs** soient **suffisamment complexes** pour résister à une attaque en force brute ;
* **surveiller les connexions** en consultant régulièrement le **fichier de journalisation /var/log/auth.log**.

## Installation et configuration de base SSH

### Installation du serveur SSH

**Remarque** …  
Le client SSH est disponible dans le **paquet openssh-client**, qui est, pour la **plupart des distributions, préinstallé**.

Presque toutes les distributions Linux incluent l'application openSSH dans le **référentiel de logiciels par défaut**.   
On peut utiliser les **utilitaires apt**, **yum|dnf** ou **zypper** pour l'installer.

Afin de **pouvoir établir une connexion à distance**, on doit installer le service SSH …  
**>> sudo apt install openssh-server**

L'installation comporte une **étape de génération des clefs de chiffrement**.  
Par la suite, le service SSH est lancé.

### Fichiers openSSH

Le paquet openSSH consiste en **plusieurs fichiers installés** …

* **Programme serveur sshd** ;
* **Programme client ssh** ;
* **Fichier de configuration serveur sshd\_conf** ;
* **Fichier de configuration client ssh\_conf** ;
* **Plusieurs fichiers de certificat** requis pour le chiffrement.

Les fichiers de configuration et de certificat sont tous **stockés dans le répertoire /etc/ssh**.   
Généralement, le processus d'installation génère également un certificat d'hôte pour identifier l'hôte de manière unique dans les connexions SSH.

**Attention** …  
Le paquet openSSH fournit à la fois les **applications serveur** (**sshd**) et **client** (**ssh**).   
En raison de la **similitude des deux programmes**, il est facile d’embrouiller les choses.   
Il en va de même pour leurs fichiers de configuration potentiels, **sshd\_conf** et **ssh\_conf**.   
il faut être prudent lors du travail avec les applications et les fichiers de configuration.

### Utilisation du serveur SSH

Le serveur SSH fonctionne en tant que **service lancé automatiquement au démarrage** de l’hôte.

Il est possible notamment de ...

* l'activer ou l'arrêter  
  si on souhaite désactiver momentanément le serveur SSH ;
* le relancer  
  si on fait une modification de configuration.

**Activation du service** ...

Pour le serveur **>> sudo systemctl start sshd.service**

Pour le client **>> sudo** **systemctl start ssh.service**

Arrêt du service ...

Pour le serveur **>> sudo** **systemctl stop sssh.service**

Pour le client **>> sudo** **systemctl stop ssh.service**

**Relance du service** (redémarrage) ...

Pour le serveur **>> sudo** **systemctl restart sshd.service**  **>> sudo** **systemctl reload sshd.service**

Pour le client **>> sudo** **systemctl restart ssh.service**  **>> sudo** **systemctl reload ssh.service**

## Première connexion avec SSH Authentification par mot de passe

L’**authentification** **par mot de passe** est la **méthode la plus simple**.

Depuis l’hôte client …  
**>> ssh <Identifiant>@<Nom DNS ou Adresse IP du serveur>  
>> ssh tux@srv01.tux.info**

Si le **même identifiant est utilisé** sur le client et sur le serveur …  
**>> ssh <Nom DNS ou Adresse IP du serveur>  
>> ssh srv01.tux.info**

Si on doit utiliser un port différent …  
**>> ssh <Nom DNS ou Adresse IP du serveur> -p <Port>  
>>** **ssh tux@srv01.tux.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 <Adresse IPv6 du serveur>  
>>** **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 ::**

**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.

## Configuration du serveur SSH

Le **fichier de configuration du serveur SSH est /etc/ssh/sshd\_config**.

**Remarque** …  
Ce fichier ne doit pas être confondu avec le **fichier /etc/ssh/ssh\_config**, qui est le **fichier de configuration du client SSH**.

En règle générale, les programmes client et serveur openSSH fonctionnent correctement tels quels après l’installation.   
Toutefois, il peut arriver que l’on souhaite personnaliser leur comportement.

Le **répertoire /etc/ssh** est **l'emplacement de stockage des fichiers de configuration** Secure Shell.   
Le **fichier de configuration /etc/ssh/sshd\_conf** définit les options du serveur au format …

**Directive Valeur**

L'**entrée directive** est le nom d'une directive à définir et l’**entrée valeur** est la valeur affectée à cette option.

**Remarque** …  
SSH n'aime pas les **paramètres contradictoires en double**.   
Par exemple, si on a ChallengeResponseAuthentication no puis ChallengeResponseAuthentication yes, **SSH respectera le premier et ignorera le second**.

Il existe **deux protocoles SSH différents numérotés 1 et 2**.   
Ce ne sont pas des versions, mais deux **protocoles distincts** développés pour fournir des connexions de données sécurisées.   
Il fut un temps où les deux protocoles étaient couramment utilisés, mais **à présent presque tous les clients SSH utilisent uniquement le protocole 2**.

Pour définir le **protocole accepté par le serveur SSH**, on utilise la directive …  
**Protocol 2**

S’il existe dans son environnement des **clients SSH plus anciens nécessitant le protocole 1**, on peut configurer son serveur SSH pour accepter les deux connexions de protocole en utilisant la directive suivante dans le **fichier /etc/ssh/sshd\_config** …  
**Protocol 1,2**

Voici les **directives les plus importantes** de ce fichier de configuration …

**Port**Cette directive indique que le **serveur SSH écoute sur le port 22**, qui est le port par défaut de SSH.   
il est **possible de faire écouter le service sur un autre port** en modifiant la valeur de cette directive.   
On peut aussi le faire **écouter sur plusieurs ports à la fois** en ajoutant des lignes similaires.  
Il est possible de modifier le serveur SSH pour écouter un autre port à l'aide de la directive Port…  
**Port 2096**

**ListenAddress**Dans certains cas, on peut avoir plusieurs interfaces réseau (ou interfaces virtuelles) et on aimerait limiter le serveur SSH afin qu'il n'écoute que certaines des cartes réseau.   
Pour ce faire, on utilise la directive ListenAddress et on spécifie l'adresse (ou les adresses) IP attribuée(s) aux interfaces réseau sur lesquelles SSH doit accepter les connexions …  
**ListenAddress 192.168.1.100:192.168.1.101**

**Attention** ...  
Si on a modifié le fichier de configuration du serveur, il faut indiquer au service de relire son fichier de configuration …  
**>> sudo systemctl reload|restart sshd.service**

**LogLevel**  
Il faudra peut-être modifier le **type de messages de journalisation** que l’on souhaite que le serveur SSH enregistre.   
Cela peut être défini à l'aide de la directive LogLevel.   
Les niveaux disponibles sont …

* **QUIET**
* **FATAL**
* **ERROR**
* **INFO**
* **VERBOSE**
* **DEBUG**
* **DEBUG1** (same as DEBUG)
* **DEBUG2**
* **DEBUG3**

Comme la journalisation du système, ces **niveaux se construisent les uns sur les autres**.   
Par exemple, le niveau **INFO inclut tous les journaux générés** aux **niveaux ERROR et FATAL** (techniquement, QUIET, mais il n’y a pas de messages à ce niveau).

**Attention** …  
Le passage à **DEBUG**, **DEBUG2** ou **DEBUG3** **peut entraîner des problèmes de sécurité** sur les serveurs SSH réels.

**Remarque** …  
La **page de manuel sshd\_config** indique:   
« *La journalisation avec un niveau DEBUG enfreint la confidentialité des utilisateurs et n'est pas recommandée*.»   
On définit la **directive LogLevel sur DEBUG** sur un hôte de test, on redémarre le serveur SSH, on se connecte au serveur SSH, puis on examine le **fichier /var/log/secure** (systèmes basés sur RHEL) ou le **fichier /var/log/auth.log** (systèmes basés sur Debian).

### Directives qui affectent l'accès utilisateur

Il est possible d’envisager de **ne pas autoriser l'utilisateur root à se connecter** directement à l’aide du protocole SSH, en particulier si le **système est connecté à Internet**. **Autoriser la connexion root** à l’aide du protocole SSH **offre aux pirates potentiels** la possibilité d’utiliser une **attaque par force brute** en essayant plusieurs mots de passe pour obtenir un accès.

Un meilleur choix consiste à **autoriser une connexion utilisateur régulière**, permettant à l'utilisateur de **basculer vers le compte root à l'aide de la commande su**. Même si l’attaque par force brute peut toujours être tentée par l’utilisateur habituel, le pirate doit savoir que le compte existe et, après avoir obtenu l’obtention d’une tentative d’accès, doit savoir le mot de passe de l’utilisateur root.

**Directive PermitRootLogin**

**Remarque** …  
Cette **directive est commentée** par défaut.

Cette directive, qui est **définie par défaut avec la valeur prohibit-password**, interdit toutes les méthodes d'authentification interactives, **n'autorisant que celle utilisant une clé publique**.   
Les valeurs peuvent être **yes**, **prohibit-password**, **forced-commands-only**, ou **no**.

Si cette option est **définie sur prohibit-password** (ou son alias obsolète without-password), le **mot de passe et l'authentification** **interactive au clavier sont désactivés** pour l’utilisateur root.

Si cette option est définie sur **forced-commands-only,** une connexion root avec **authentification par clé publique est autorisée**, mais uniquement **si l'option de commande a été spécifiée** (ce qui peut être utile pour effectuer des sauvegardes à distance même si la connexion root n'est normalement pas autorisée). Toutes les **autres méthodes d'authentification sont désactivées** pour l’utilisateur root.

Si cette option est définie sur **no**, root n'est pas autorisé à se connecter.

Si cette **directive est égale à no**, ce qui signifie que root ne peut se connecter à distance, on doit d'abord vous con**necter par SSH en tant que simple utilisateur**, puis utiliser la commande su pour **basculer à root**. Sans cela, un pirate n'aurait qu'à trouver le mot de passe du compte root, alors que là, il doit trouver un compte d’utilisateur et son mot de passe.

**Directive AllowUsers et DenyUsers**

Il est aussi possible de limiter seulement certains utilisateurs pouvant se connecter à l’aide de SSH  
Dans ce cas, il existe deux choix … **AllowUsers** ou **DenyUsers**.

Ces deux mots clés vous permettent de spécifier une liste de noms d'utilisateurs …  
**AllowUsers tux kermit squiddly**

* Si on utilise le **directive AllowUsers**, seuls ces utilisateurs seront autorisés à se connecter au serveur SSH ;
* Si on utilise le **directive DenyUsers**, tous les utilisateurs peuvent se connecter au serveur SSH, à l'exception des utilisateurs répertoriés.

**Remarque** …  
Si on utilise les deux directives, **DenyUsers est prioritaire**.

Il est possible d’utiliser des caractères génériques pour faire correspondre les modèles …  
**DenyUsers invit\* # Refuser à tout utilisateur portant un nom commençant par « invit »  
DenyUsers ????app # Refuser à tout utilisateur avec un nom qui contient 4 caractères et qui se termine par « app »**

**Remarque** …  
Il existe également une **directive AllowGroups** et une **directive DenyGroups**.

Le tableau suivant répertorie les options les plus courantes que l’on devra peut-être modifier sur le serveur openSSH.

|  |  |
| --- | --- |
| Directive | Description |
| Banner | Spécifie le fichier à afficher avant la connexion d’un usager |
| AllowUsers | Autorise la liste d'utilisateurs spécifiée à se connecter à l'aide de SSH |
| DenyUsers | Empêche la liste d'utilisateurs spécifiés de se connecter à l'aide de SSH |
| PasswordAuthentication | Autorise l'authentification à l'aide de mots de passe textuels |
| MaxAuthTries | Spécifie le nombre de tentatives qu’un utilisateur peut entrer un mot de passe (défaut 6) |
| PermitRootLogin | Autorise le compte utilisateur root à se connecter à l'aide de SSH |
| PermitEmptyPassword | Spécifie si un usager peut se connecter sans mot de passe (défaut No) |
| PrintMotd | Spécifie si le contenu du fichier /etc/motd est affiché lorsqu’un usager se branche avec SSH |
| Protocol | Spécifie le niveau de protocole de chiffrement – Le niveau 2 est préféré et plus sécurisé |
| PubkeyAuthentication | Autorise l'authentification à l'aide de certificats |
| X11Forwarding | Autorise les serveurs X distants à exécuter des applications client X à l'aide d'un tunneling |
| AllowTcpForwarding | Autorise le serveur à accepter les protocoles en tunnel |
| listenAddress | Spécifie les interfaces qu’écoute SSH |
| Port | Modifie le serveur SSH pour écouter un autre port |

La plupart de ces directives prennent une **simple valeur oui ou non** pour activer ou désactiver la fonctionnalité.   
Les **directives AllowUsers et DenyUsers** utilisent une **liste de noms de compte d'utilisateur** sur le système, **séparés par des espaces**, afin de déterminer les utilisateurs à autoriser ou à bloquer l'accès au serveur.

Le **fichier de configuration sshd\_conf** par défaut **contient de nombreuses autres options**, décrites dans les commentaires qu'il contient. Il est important de de prendre le temps de parcourir chaque option pour déterminer s’il s’agit d’un avantage pour son environnement réseau.

**Vérification du fichier sshd\_cinfig**

On peut vérifier la configuration à l’aide de la commande avec sshd -T et vérifier la sortie …  
**>> sudo sshd –T**  
**etc/ssh/sshd\_config: line 7: Bad configuration option: bonjour**  
**/etc/ssh/sshd\_config: terminating, 1 bad configuration options**

Après avoir **apporté des modifications à la configuration**, il faudra **redémarrer le serveur openSSH** pour que celles-ci prennent effet.   
Il est également possible de **simplement recharger le fichier de configuration**, cela permettra de ne pas déconnecter les utilisateurs utilisant le service.

Pour ce faire, on utilise …  
**>> sudo systemctl reload|restart sshd.service**

## Utilisation d’openSSH

Il est intéressant de connaître plusieurs commandes de client SSH …

* **ssh**  
  Cette commande permet de se connecter à un serveur SSH et de travailler sur le serveur à l’aide d’une ligne de commande ;
* **scp**  
  Cette commande permet de transférer des fichiers vers et/ou depuis un serveur SSH à l’aide de la ligne de commande ;
* **sftp**  
  Cette commande permet de se connecter à un serveur SSH et d'exécuter des commandes de type **ftp**.

On doit **connaître la fonction** de chacune de ces commandes ainsi que certaines **directives clés du fichier /etc/ssh/ssh\_config**.

### Fichier ssh\_config

Quelques **directives du fichier ssh\_config sont différentes du fichier sshd\_config** (voir plus loin).   
Pour commencer, il y a le **fichier système /etc/ssh/ssh\_config**, qui s’applique à **tous les utilisateurs**.   
De plus, **chaque utilisateur peut créer un fichier dans son répertoire personnel** (~/.ssh/config), qui peut être utilisé pour remplacer les directives du fichier /etc/ssh/ssh\_config.

Outre les fichiers, les **options de ligne de commande peuvent remplacer les valeurs spécifiées** dans les fichiers de configuration.   
L'ordre dans lequel toutes ces informations sont analysées est …

* **Options de ligne de commande** ;
* **Fichier ~/.ssh/config de l'utilisateur** ;
* **Fichier /etc/ssh/ssh\_config** (fichier global).

**Important** …  
La **première directive trouvé est celle utilisée.**   
Par exemple, si **ConnectTimeout** est défini dans le fichier ~/.ssh/config de l'utilisateur et qu'une valeur différente est définie dans le fichier /etc/ssh/ssh\_config, le fichier de configuration de l'utilisateur est utilisé pour définir cette valeur.

Une autre **différence importante entre le fichier ssh\_config et le fichier sshd\_config** est que la plupart des directives du fichier ssh\_config sont des **sous-direcitves de la directive Host**.   
La **directive Host** permet de **spécifier différentes règles pour différents serveurs SSH** auxquels on se connecte.   
Par exemple, les éléments suivants appliqueraient la valeur 0 de **ConnectTimeout** lors de la connexion à srv01.profsavard.com et la valeur 600 lors de la connexion à test.example.org …

**Host srv01.profsavard.info  
 ConnectTimeout 0  
Host test.example.org  
 ConnectTimeout 600**

**Remarque** …  
La **plupart des** **directives du fichier ssh\_config sont liées aux directives du fichier de configuration du serveur SSH**.

### Commande ssh

La **première fois que l’on essaie de se connecter** à un hôte à l’aide de la commande ssh, on est **invité à vérifier l'empreinte de la clé RSA** …  
**>> ssh** [**tux@srv01.tux.info**](mailto:tux@srv01.tux.info) **The authenticity of host 'srv01.monserveur.info (192.168.21.100)' can't be established.  
ECDSA key fingerprint is SHA256:VYhBBhvTXSwfT84uso27yc2kf5FV8iaqzPv8uwA+dlo.  
Are you sure you want to continue connecting (yes/no)? yes**

Cette clé sera stockée dans le répertoire personnel de l’utilisateur …   
**>> sudo** **cat .ssh/known\_hosts  
|1|eyUODv8CE6lMPwjL7NEuSQWj9zY=|7KkYqaMJpNO6hS2vk+qLVmkyBh8= ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBL4VxsY+jio+La0yg2huP6XYde9qdU86bX2Ju4D8ThY+RQJoqD9BFyevbBNcsx3zEjgvx49wVJaXr+iepj1WPX8=**

En tant qu’utilisateur, on n’a pas **besoin de connaître précisément les empreintes numériques** ; ce processus étant géré par l'utilitaire client SSH et le serveur SSH.   
Toutefois, il s’agit d’une **situation concernant l’empreinte numérique** que l’on doit connaître.   
Elle est liée au **message suivant** …

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
@ 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  
7a:9f:9b:0b:7b:06:3a:f0:97:d1:c7:e9:94:a8:84:03  
Please contact your system administrator.**

Ce message d'erreur **peut apparaître pour plusieurs raisons** …

* Le **logiciel du serveur SSH a été réinstallé**.   
  Cela pourrait impliquer une nouvelle empreinte de clé RSA ;
* Le **serveur SSH lui-même a été réinstallé ou remplacé** par un nouveau système ;
* L’**attaque** de l'homme au milieu (*man-in-the-middle*) pourrait en être la cause.

L'administrateur du serveur SSH doit savoir si l'une des deux premières situations s'est produite.  
 Si tel est le cas, la solution consiste à **supprimer l'entrée de ce serveur SSH du fichier ~ /.ssh/known\_hosts**.   
Si les deux premières situations ne se sont pas produites, il est temps de **faire quelques recherches pour déterminer ce qui se passe réellement**.

### Connexion à un serveur SSH

Avec le serveur openSSH en cours d'exécution, on **peut s’y connecter à l'aide du programme client ssh**.   
Afin de se connecter au serveur distant en utilisant le même compte d'utilisateur que l‘on utilise sur le client, on spécifie simplement le nom d'hôte ou l'adresse IP du serveur distant …

Depuis l’hôte client …  
**>> ssh <Identifiant>@<Nom DNS ou Adresse IP du serveur>  
>> ssh tux@srv01.profsavard.info  
>> ssh tux@192.168.1.100**

Si le **même identifiant est utilisé** sur le client et sur le serveur …  
**>> ssh <Nom DNS ou Adresse IP du serveur>  
>> ssh srv01.profsavard.info**   
**>> ssh tux@192.168.1.100**

Cette commande tente de se connecter à l'hôte distant 192.168.1.100 avec l'ID utilisateur tux.

* Si le compte d'utilisateur est défini pour l'**authentification par mot de passe**, on sera invité à entrer son mot de passe sur le système distant ;
* Si le compte d'utilisateur est configuré pour l'**authentification** **par certificat**, une invite de commande apparaîtra immédiatement sur le système distant.

Si on doit se **connecter sous un autre compte utilisateur**, on l’indique simplement sur la ligne de commande à l'aide du signe esperluette (@), comme une adresse électronique …  
**>> ssh -l <Nom d’utilisateur>@<Nom DNS ou Adresse IP du serveur>  
>> ssh -l kermit@srv01.profsavard.info**   
**>> ssh -l kermit@192.168.1.100**

Pour exécuter une commande sur le système distant, mais revenir immédiatement à son système client, on utilise la syntaxe …  
**>> ssh <Nom DNS ou Adresse IP du serveur> <Commande>  
>> ssh 192.168.1.100 "apt update && apt upgrade -y"**    
**>> ssh 192.168.1.100 "systemctl status sshd.service"**

### Commandes scp et sftp

La commande scp utilise plusieurs des **mêmes options et fonctionnalités fournies par la commande ssh**.   
Pour copier un fichier de l’hôte actuel vers un hôte distant, on utilise la syntaxe suivante …  
**>> scp <Nom du fichier> <Nom DNS ou Adresse IP du serveur>:<Répertoire>  
>> scp /etc/hosts 192.168.1.22:/tmp**  
**>> scp monfichier.texte tux@192.168.1.77:/home/tux/Documents/monfichier.texte**

Pour transférer un fichier depuis l’hçote distant vers le système local, on utilise …  
**>> scp tux@192.168.1.77:/home/tux/Documents/monfichier.texte monfichier.texte**

Une fonctionnalité pratique de scp est la possibilité de copier des fichiers entre deux serveurs distants …  
**>> scp** tux**@192.168.1.100:monfichier.texte tux@192.168.1.101:monfichier.texte**

Quelques **autres options de la commande scp utiles** …

* **-r**  
  Copie des structures de répertoires entières ;
* **-v**  
  Mode verbeux  
  Affiche des détails utiles pour le débogage des problèmes de transfert ;
* **-q**Mode silencieux ;
* **-p**  
  Préserve les horodatages et les autorisations.

La commande sftp se connecte à un serveur SSH et fournit une interface client de type ftp.

**>> scp** [**tux@srv01.profsavard.info**](mailto:tux@srv01.profsavard.info) **tux@srv01.profsavard.info's password:  
Connected to 192.168.1.13.  
sftp> pwd  
Remote working directory: /home/tux  
sftp> ls  
tux.texte  
sftp> get tux.texte .  
Fetching /home/tux/tux.texte to ./tux.texte  
sftp> ls  
tux.texte  
sftp> bye  
>>**

## Fonctionnalités SSH avancées

### Connexion sans mot de passe

Si un utilisateur se connecte régulièrement à un serveur SSH spécifique, il peut vouloir **configurer des connexions sans mot de passe**.

Les étapes à suivre sont …

**Étape 1**Sur l’hôte client SSH, on exécute la commande ssh-keygen.   
Lorsque l’on est invité à entrer une phrase secrète, on appuie sur la touche Entrée pour laisser ce champ vide (ou on peut utiliser l'option -N avec la commande ssh-keygen, ce qui entraîne l'absence d'invite de phrase secrète) …  
**>> ssh-keygen -t rsa  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/tux/.ssh/id\_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/tux/.ssh/id\_rsa.  
Your public key has been saved in /home/tux/.ssh/id\_rsa.pub.  
The key fingerprint is:  
SHA256:cB/O6k0Wh/0KK/PycdoWhca0WIeWbLNxo6tIsTWVX98** [**root@srv01.profsavard.info**](mailto:root@srv01.profsavard.info) **The key's randomart image is:  
+---[RSA 2048]----+  
| . + |  
| & + .|  
| . . .O @ oo|  
| o.+++B o E|  
| S+\*ooo |  
| o. oo. |s  
| ...=.... |  
| .=+.B.. |  
| .\*\*.o |  
+----[SHA256]-----+**

Pour confirmer la présence des clés …  
**>> ls ./.ssh/  
id\_rsa id\_rsa.pub**

Pour afficher le contenu de la clé publique …  
**>> cat ./.ssh/id\_rsa.pub   
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDMbVkbxyTVo8ZCX4iJC3TBoEK2CNnlOax9LpmuQDq6iTK4FCGy1jovCjY7+/9/rvDeewVCDCn0mbvfOi/9P1mo7Pz86uBrhjwyhGsN+mGottcn+zxBOZS/6P6O/M/O71WOKNBWfDbxXdLxcCOhHvqx0f+hfGawGtgrF9xT42BvACWHqhqCiRIf4uzd69GapBP2w2FqEPVpauRjpkPH6BJNz2vhJf+jv4mJEsWsA52Zj5JOrVmSybjyryhr2ner4Pef6pd2/br1Ht2wHrmskZOF2SB3fdzylJvuyNnr0Fm+dQNVp9rc3hphMnWd5a6zakNB51yh28SxbURX9BF4ENb9 lsavard@debian-cli**

**Étape 2**Il existe maintenant une **clé publique dans le fichier ~/.ssh/id\_rsa.pub**.

Celui-ci **doit être copié sur le serveur distant dans le fichier ~/.ssh/registered\_keys** de son **compte sur l’hôte distant**.  
On devra peut-être créer ce répertoire et définir les autorisations sur 770).

Cela peut être fait avec les commandes suivantes …  
**>> cat .ssh/id\_rsa.pub | ssh -l tux srv01.tux.info 'cat >> .ssh/authorized\_keys'  
tux@srv01.profsavard.info's password:**

La clé publique va se retrouver dans le **fichier /home/lsavard/.ssh/authorized\_keys** sur le serveur.

**Étape 3**On définit les **autorisations du fichier allowed\_keys sur 640** …  
**>> ssh -l tux srv01.profsavard.info 'chmod 640 /home/tux/.ssh/authorized\_keys'  
tux@ srv01.profsavard.info 's password:**

Il est **maintenant possible de se connecter au serveur SSH sans mot de passe** …  
**>> ssh -l tux srv01.profsavard.info 'date'  
Linux debian-cli 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) 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: Mon May 24 15:05:36 2021 from 192.168.1.104**

Certains experts en sécurité soulignent que **le fait de ne pas utiliser de phrase secrète pour sa clé présente un risque** pour la sécurité. Cependant, **si on a une phrase secrète**, **chaque fois que l’on essaie de se connecter** au serveur SSH à partir du client, au lieu de demander le mot de passe, **on sera invité à entrer la phrase secrète** (ce que l’on tentait d'éviter en premier lieu) …

**Étape 1**on **démarre un nouveau shell bash avec l'utilitaire ssh-agent** …  
**>> ssh-agent /bin/bash**

**Étape 2**On **exécute la commande suivante** …  
**>> ssh-add ~/.ssh/id\_rsa**

À partir de ce moment, lorsque l’on **utilise le shell ssh-agent** pour se connecter à distance au serveur SSH, la **connexion a lieu sans mot de passe**.

### Suppression des clés autorisées du serveur

Il suffit de supprimer du fichier /home/$USER/.ssh/authorized\_keys la clé qui correspond à l’utilisateur dont l’accès a été révoqué. Après avoir supprimé la clé, on sauvegarde le fichier.

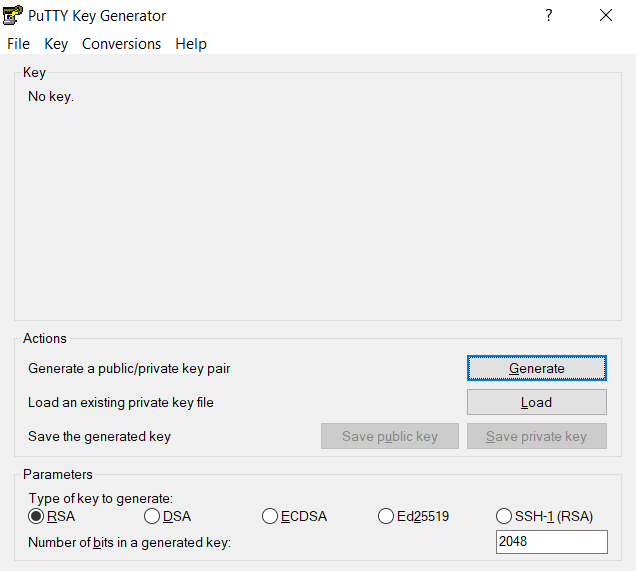
## Gestion des clés à l’aide de Putty

### Création une clé SSH à l’aide de PuTTY (sous Windows)

PuTTY est un client SSH répandu pour Windows.   
On peut l’utiliser pour se connecter à distance à un serveur Linux.   
Son logiciel compagnon, **Putty Key Generator** ou **PuTTYgen**, peut être **utilisé pour créer des clés SSH**.

Tout d’abord, si ce n’est déjà fait, il faut **télécharger le logiciel PuTTYgen**, qui **servira à générer la clé**.

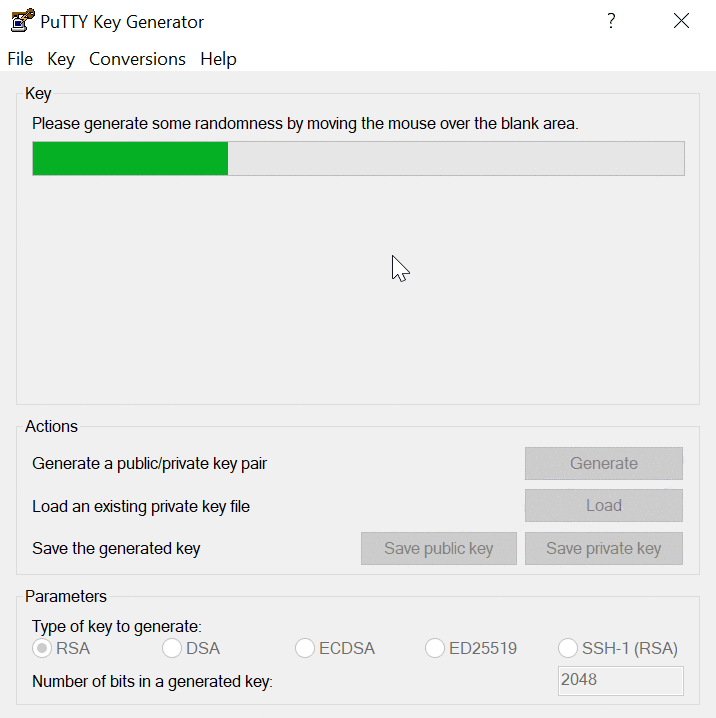
On exécute ensuite l’utilitaire.



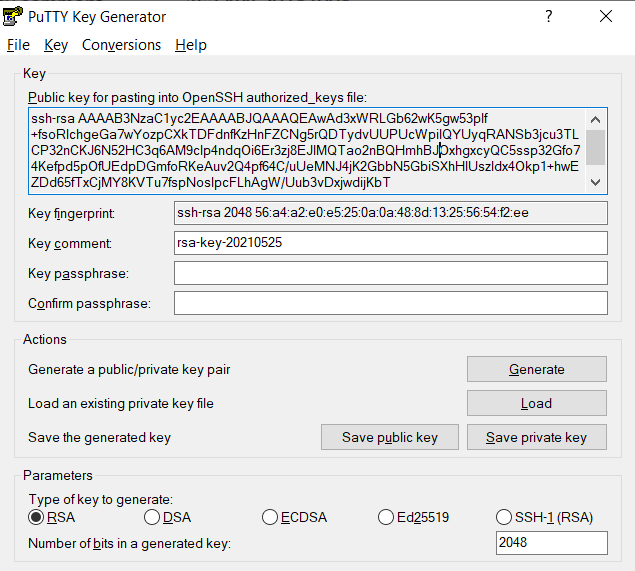
Dans un premier temps, on choisit le **type de clé.**   
Dans cet exemple, il s’agit d’une **clé RSA de 2048 bits**.

On clique sur **Générer** pour **lancer le processus de création**.

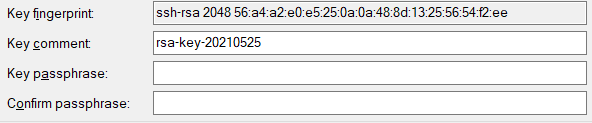
On **déplace le curseur de la souris** de façon aléatoire dans la zone située sous la barre de progression **afin d’accélérer la création** de la clé.



**PuttyGen**, à la **fin du processus de création**, affiche ce qui suit …



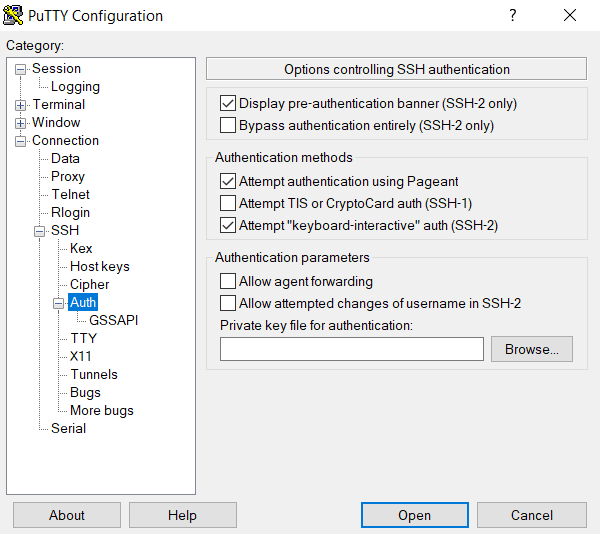
Bien que **cela ne soit pas obligatoire** on peut saisir, par 2 fois, un **mot de passe**, qu'il faudra **noter ou retenir**.  
Ainsi celui qui arrive à copier cette clé privée ne pourra pas s'en servir sans saisir ce mot de passe.



Il faut enfin sauvegarder les clés publique et privée.

### Utilisation de la clé publique avec Putty

Afin d’utiliser sa clé pour **ouvrir une session SSH avec Putty** …  
**Connection > SSH > Auth** et on indique le **chemin sous la rubrique Private Key file for authentification**.



### Suppression des clés autorisées du serveur

Il suffit de supprimer du fichier /home/$USER/.ssh/authorized\_keys la clé qui correspond à l’utilisateur dont l’accès a été révoqué. Après avoir supprimé la clé, on sauvegarde le fichier.

# En vrac

## 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.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**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**

# Paramètres de sshd\_conf

Précisions sur les paramètres su fichier sshd\_config …

* **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.
* **Subsystemsftp**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.