# Antisèches Informatiques

« Mapper un serveur ftp dans l'explorateur Windows - Travailler sur un grand nombre de fichiers sous »

# Configuration et utilisation de openssh

Par Rémi TAUPIN le vendredi 23 janvier 2009, - Lien permanent

commandes, debian, etch, <u>firefox</u>, <u>lenny</u>, <u>linux</u>, macosx, proxy, <u>reseaux</u>, <u>securite</u>, <u>squeeze</u>, <u>ssh</u>, systeme, windows



**Openssh** fournit une suite d'outils clients et serveurs qui permettent de faire des échanges sécurisés entre des machines à travers un réseau non sécurisé. Openssh est à la fois un protocole réseau et une suite d'exécutables tels que **sshd** et **sftpd** pour les logiciels serveurs, et **ssh**, **sftp** ou encore **scp** pour les logiciels clients.

Dans une utilisation de base, openssh permet de remplacer **telnet** et les outils **rlogin**, mais on peut également l'utiliser pour monter des tunnels vpn.

Cet article présente l'installation et l'utilisation de openssh sous Linux, Windows et MacOSX en utilisant une authentification simple par mot de passe puis une authentification grâce à une paire de clés privée/publique.

Il présente également la création de tunnels sécurisés et le montage de partitions distantes avec sshfs.

# Sommaire

Openssh pour linux

```
(http://www.antiseches.net/post/Configuration-et-utilisation-de-openssh#opensshlin)
Openssh pour macosx (http://www.antiseches.net/post/Configuration-et-utilisation-
de-openssh#opensshosx)
Openssh pour windows (http://www.antiseches.net/post/Configuration-et-utilisation-
de-openssh#opensshwin)
Putty pour windows (http://www.antiseches.net/post/Configuration-et-utilisation-
de-openssh#puttywin)
Création de tunnels sécurisés (http://www.antiseches.net/post/Configuration-
et-utilisation-de-openssh#tunnels)
Résumé des commandes de base (http://www.antiseches.net/post/Configuration-
et-utilisation-de-openssh#cdesbase)
```

# Openssh pour linux

## Installation

Avec une distribution Debian, il est trés simple d'installer openssh :

homeserver:~# apt-get install ssh

La procédure d'installation dure quelques secondes au cours desquelles le serveur et le client ssh sont installés. Une fois terminée, tout est prêt pour pouvoir commencer à

travailler. La syntaxe pour se connecter à un serveur distant est :

```
client:~# ssh -l user serveur.exemple.com
OU
client:~# ssh user@serveur.exemple.com
```

Si l'utilisateur local et distant ont le même nom, il n'est pas utile de spécifier **user**. La ligne de commande devient alors :

```
client:~# ssh serveur.exemple.com
```

Lors de la première connexion à un serveur, le client ssh avertit que l'identité de ce serveur ne peut pas être vérifiée. C'est normal car il ne le connait pas encore. Il demande alors une confirmation :

```
The authenticity of host 'serveur.exemple.com (192.168.0.1)' can't be
established.
RSA key fingerprint is fa:0a:7e:04:bc:19:19:67:e1:46:51:eb:12:c4:5b:97.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'client.domaine.com' (RSA) to the list of known
hosts.
Password: *******
client:~# exit
```

Après confirmation, le client ssh ajoute l'empreinte de la clé (**key fingerprint**) dans le fichier ~/.ssh/known\_hosts de la machine cliente client.domaine.com. Par exemple si l'utilisateur est root et que le PC client fonctionne sous linux, le fingerprint du serveur sera ajouté au fichier "/root/.ssh/known\_hosts".

On peut maintenant se connecter au serveur distant avec la commande ssh root@serveur.exemple.com (si on veut utiliser le compte root).

```
client:~# ssh root@serveur.exemple.com
Password: *******
Last login: Sun Dec 10 15:17:44 2006 from 192.168.0.10
client:~#
```

# Authentification par bi-clés

Si on se connecte fréquemment sur ce serveur, il peut être pénible de rentrer le mot de passe à chaques fois. On va donc créer une paire de clés privée/publique et on va configurer le serveur pour qu'il nous reconnaisse automatiquement grâce à notre clé publique et ne nous demande plus de mot de passe.

Pour ce faire on utilise la commande ssh-keygen :

client:~# ssh-keygen -t rsa -b 1024 -C "Cle\_rsa\_1024\_bits\_de\_root" Generating public/private rsa key pair. Enter file in which to save the key (/home/root/.ssh/id\_rsa): [Enter] Enter passphrase (empty for no passphrase):[Enter] Enter same passphrase again:[Enter] Your identification has been saved in /home/root/.ssh/id\_rsa. Your public key has been saved in /home/root/.ssh/id\_rsa.pub. The key fingerprint is: 1a:2a:3e:4a:1a:aa:cb:29:f3:1f:a1:06:c7:8e:e9:58 Cle\_rsa\_1024\_bits\_de\_root

#### Explication de la commande :

ssh-keygen : commande pour générer une paire de clés

- -t rsa: on veut une clé de type rsa
- -b 1024 : la longeur de la clé est de 1024 bits
- -c "cle\_rsa...." : c'est un commentaire optionnel pour identifier la clé. Il sera utile

pour l'identifier dans le fichier **~/.ssh/authorized\_keys** (voir plus loin pour l'explication de ce fichier).

Note au sujet de la passphrase qui est demandée : cette passphrase et une sécurité supplémentaire au cas où la clé privée serait dérobée. Le problème est que même si elle augmente la sécurité du système, elle en complique l'utilisation dans le sens où elle oblige à mettre en place un autre mécanisme, **ssh-agent** sur le serveur distant. On la laissera donc vide.

Maintenant que les clés sont générées, il faut copier la clé publique sur le serveur distant. On peut le faire en utilisant l'utilitaire de copie sécurisée **scp** qui fait partie des outils fournis en standard par openssh :

```
client:~# scp -P 22 /root/.ssh/id_rsa.pub root@serveur.exemple.com:/root
/.ssh/idclient_rsa.pub
Password: ******
/root/.ssh\i 100% 230 0.2KB/s 00:00
```

Si le serveur ssh distant fonctionne sur le port tcp par défaut (le port 22) on peut omettre le paramètre -P 22.

<u>/!\</u> Attention, dans la commande ci-dessus le fichier de destination est bien idclient\_rsa.pub et non pas id\_rsa.pub. En effet un fichier id\_rsa.pub existe surement et on risque de l'écraser. Et celui qui est déjà présent est certainement la clé publique rsa de l'administrateur du serveur. Ce serait dommage de l'écraser...

Maintenant que la clé publique est copiée, on ajoute son contenu au fichier /root/.ssh /authorized\_keys du serveur. Toujours à partir de la machine client, on utilise les commandes suivantes :

```
client:~# ssh root@serveur.exemple.com "cat /root/.ssh/idclient_rsa.pub >>
/root/.ssh/authorized_keys"
Password: *******
```

Comme dans la commande précédente, si le serveur ssh distant fonctionne sur le port tcp par défaut (le port 22) on peut omettre le paramètre -p 22. Notons que dans la commande scp on écrit -p 22 (P majuscule) alors que pour la commande ssh on écrit -p 22 (p minuscule).

<u>/!\</u> Attention, dans la commande ci-dessus il faut bien mettre >> et non pas > sinon on écrase le fichier **authorized\_keys** du serveur distant au lieu de le mettre à jour...

Si tout a fonctionné, on peut maintenant se connecter sans être obligé de rentrer le mot de passe :

```
client:~# ssh root@serveur.exemple.com
Last login: Sun Dec 10 20:18:09 2006 from 192.168.0.10
serveur:~#
serveur:~#
```

Ca marche, le serveur distant ne demande pas de mot passe !

Si on se connecte souvent sur ce serveur, il peut être pénible de rentrer à chaques fois la commande complète ssh user@serveur.exemple.com. Pour simplifier la ligne de commande, il suffit de créer ou de modifier le fichier ~/.ssh/config de la façon suivante :

```
Host serveur
Port 22
User root
HostName serveur.exemple.com
```

A partir de maintenant, un simple ssh serveur suffira pour ce connecter sur la machine **serveur** en question. Le paramètre **Port 22** est facultatif si le serveur fonctionne sur le port par défaut.

Nous pouvons maintenant aller voir plus bas pour un résumé des <u>commandes de base</u> (http://www.antiseches.net/#cdebases) ou pour apprendre à gérer des <u>tunnels ssh</u> (http://www.antiseches.net/#tunnels).

# Openssh pour MacOSX

MacOSX est maintenant un beau système d'exploitation Unix Like puisqu'il est basé sur FreeBSD 4.4. C'est d'ailleurs un des systèmes Unix le plus utilisé dans le monde.

Sous MacOS 10.5, openssh est déjà installé en standard et il fonctionne exactement comme sous linux.

Pour l'utiliser en ligne de commande, il faut ouvrir un shell en allant dans Finder / Applications / Utilitaires et en cliquant sur l'icone Terminal.

Ensuite, comme sous linux on utilise la commande ssh pour se connecter à un serveur distant.

Le démon **sshd** de MacOSX est présent mais n'est pas lancé au démarrage. Pour le lancer manuellement, il faut utiliser la commande <u>service ssh</u> start. On peut alors se connecter à la machine avec un client ssh.

# **Openssh pour Windows**

Openssh pour Windows est le portage win32 de openssh. Il s'utilise exactement comme sous linux. Il faut cependant noter que sa version n'a pas évolué depuis longtemps, contrairement à la version pour linux.

# Téléchargement du logiciel

```
http://sshwindows.sourceforge.net/ (http://sshwindows.sourceforge.net/)
http://sshwindows.sourceforge.net/download/ (http://sshwindows.sourceforge.net
/download/)
```

## Installation de openssh pour win32

Faire un double-clic sur l'icone du programme d'installation et suivre les étapes suivantes :



Lire et accepter les termes du contrat :



Choisir les composants à installer. Le serveur ssh ne sera sélectionné que si on désire se connecter en ligne de commande sur cette machine :

OpenSSH for Windows 3	.8.1p1-1 Setup	
Choose Components Choose which features of Oper	nSSH for Windows 3.8. Ip1-1 you	u want to install.
Check the components you wa install. Click Next to continue.	nt to install and uncheck the com	ponents you don't want to
Select components to install:	<ul> <li>✓ Shared Tools</li> <li>✓ Client</li> <li>✓ Server</li> <li>✓ Start Menu Shortcuts</li> </ul>	Description Hover your mouse over a component to see its description.
Space required: 4.7MB		
Nulsoft Install System v2.0	< Back	Next > Cancel

En général, on laisse toutes les options par défaut :

OpenSSH for Windows 3.8.1p1-1 Setup	20
Choose Start Menu Folder Choose a Start Menu folder for the OpenSSH for Windows 3.8.1p1-	1 shortcuts.
Select the Start Menu folder in which you would like to create the pr can also enter a name to create a new folder.	ogram's shortcuts. You
7-Zip Accessories Accessories Belkin CDBurnerXP Pro 3 Démarrage Inter Wideo WinDVD Jeux Microsoft J.NET Framework SDK v2.0 Microsoft Office Microsoft Office Microsoft Office Microsoft Vindows Mobile 5.0 MSPP Emulator Images Multimédia Nulsoft Linstal System v2.0	



Si on a choisit d'installer la partie server d'open ssh, il faudra importer et configurer les comptes des utilisateurs qui seront autorisés à ce connecter sur cette machine. Le programme d'installation montre quel est le fichier qui explique la marche à suivre :

	OpenSSH for Windows 3.8.1p1-1 Setup	
	Installing Please wait while OpenSSH for Windows 3.8. 1p1-1 is being installed.	
	Execute: C:\Program Files\OpenSSH\bin\ssh-keygen.exe -t rsa -f /etc/ssh_host_rsa_key -N **	
Open!	SSH for Windows 3.8.1p1-1 Setup	×
٩	Before starting the OpenSSH service you MUST edit the C:\Program Files\OpenSSH\etc\passwd file. If you this, you will not be able to log in through the SSH server. Please read the readme.txt or quickstart.txt fil information regarding proper setup of the passwd file.	u don't do e for
	ОК	
	Nullsoft Install System v2.0	

L'installation est maintenant terminée :

OpenSSH for Windows	3.8.1p1-1 Setup
	Completing the OpenSSH for Windows 3.8.1p1-1 Setup Wizard OpenSSH for Windows 3.8.1p1-1 has been installed on your computer. Click Finish to close this wizard.
	< Back Einish Cancel

Sans plus de paramètrages, le client openssh fonctionne déjà. Pour l'utiliser, il suffit d'ouvrir une **Invite de Commande** :



Lors de la première connexion à un serveur, le client ssh avertit que l'identité de ce serveur ne peut pas être vérifié, ce qui est normal car il ne le connait pas encore. Il demande alors une confirmation.

Après confirmation, le client ssh ajoute l'empreinte de la clé (**key fingerprint**) dans le fichier %profile%\.ssh\known\_hosts de la machine cliente. Par exemple si l'utilisateur est "Administrateur" et que le PC tourne sous Windows XP, le fingerprint sera dans le fichier "C:\Documents and Settings\Administrateur\.ssh\known\_hosts" :

```
C:\>ssh user@serveur.exemple.com

The authenticity of host 'serveur.example.com (192.168.0.1)' can't be

established.

RSA key fingerprint is a8:9a:7e:04:ab:19:19:67:e4:36:52:eb:12:f4:50:a7.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added 'serveur.example.com, 192.168.0.1' (RSA) to the

list of known hosts.

Password: *******

serveur:~# exit
```

Lors de la deuxième connexion au serveur ssh, l'identité du serveur sera vérifiée et si elle est confirmée, il n'y aura pas de question supplémentaire :

```
C:\>ssh user@serveur.exemple.com
Password: *******
Last login: Sun Dec 10 15:17:44 2006 from 192.168.0.10
serveur:~#
```

#### Authentification par bi-clés

Si on se connecte fréquemment sur ce serveur, il peut être pénible de rentrer le mot de passe à chaques fois. On va donc créer une paire de clés privée/publique et on va configurer le serveur pour qu'il nous reconnaisse grâce à notre clé au lieu du mot de passe.

Pour ce faire on utilise la commande ssh-keygen :

```
C:\>ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/Administrateur/.ssh/id_rsa):
[Enter]
Enter passphrase (empty for no passphrase):[Enter]
Enter same passphrase again:[Enter]
Your identification has been saved in /home/Administrateur/.ssh/id_rsa.
Your public key has been saved in /home/Administrateur/.ssh/id rsa.pub.
```

The key fingerprint is: la:2a:3e:4a:la:aa:cb:29:f3:lf:al:06:c7:8e:e9:58 Administrateur@machinecliente

Comme pour openssh pour linux, cette commande crée une paire de clés dsa. Avec la version win32 de openssh, on ne peut pas mettre d'autres paramètres que -t rsa sur la ligne de commande sinon on a un message d'erreur.

Si on lit la sortie générée par la commande précédente, on voit qu'elle a créé les 2 fichiers suivants :

- La clé publique RSA **id\_rsa.pub** de l'utilisateur (ici administrateur) dans le répertoire **/home/Administrateur/.ssh/** 

- La clé privée RSA **id\_rsa** de l'utilisateur (toujours administrateur) dans le répertoire/**home/Administrateur/.ssh**/

En fait ce n'est pas tout à fait vrai : les clés ont été créées dans **C:\Documents and Settings\Administrateur\.ssh** au lieu de **/Home/Administrateur**, mais le message de sortie de la commande est faux. Il a été fait pour linux et n'a pas été modifié lors de son portage pour win32...

Maintenant que les clés sont générées, il faut copier la clé publique sur le serveur distant. On peut le faire en utilisant l'utilitaire de copie sécurisée **scp** qui fait partie des outils fournis en standard par openssh :

C:\>scp "\Documents and Settings\Administrateur\.ssh\id\_rsa.pub" root@serveur.exemple.com:/root/.ssh/idclient\_rsa.pub Password: \*\*\*\*\*\* \Documents and Settings\Administrateur\.ssh\i 100% 230 0.2KB/s 00:00

Attention, dans la commande ci-dessus le fichier de destination est bien **idclient\_rsa.pub** et non pas **id\_rsa.pub**. En effet un fichier **id\_rsa.pub** existe surement et on risque de l'écraser. Et celui qui est déjà présent est certainement la clé publique rsa de l'administrateur du serveur...

Maintenant que la clé est copiée, on ajoute son contenu au fichier /root/.ssh /authorized\_keys :

```
C:\>ssh root@serveur.exemple.com "cat /root/.ssh/idclient_rsa.pub >>
/root/.ssh/authorized_keys"
Password: *******
```

Attention, dans la commande ci-dessus il faut bien mettre >> et non pas > sinon on écrase le fichier **authorized\_keys** du serveur distant au lieu de le mettre à jour...

Si tout a fonctionné, on peut maintenant se connecter sans être obligé de rentrer le mot de passe :

```
C:\>ssh root@serveur.exemple.com
Last login: Sun Dec 10 20:18:09 2006 from 192.168.0.10
serveur:~#
serveur:~#
```

Ca marche !

On peut maintenant effacer la clé du client sur le serveur :

C:\>ssh root@serveur.exemple.com "rm /root/.ssh/idclient\_rsa.pub" C:\>

Bien sûr aucun mot de passe n'est demandé et il n'y a pas de demande de confirmation lors de l'effacement du fichier sur le serveur distant. Prudence donc...

Si on se connecte souvent sur ce serveur, il peut être pénible de rentrer à chaques fois la commande complète <u>ssh</u> <u>root@serveur.exemple.com</u>. Pour simplifier la ligne de commande, il suffit de créer ou de modifier le fichier **C:\Documents** and **Settings\Administrateur\.ssh\config** de la façon suivante :

Host serveur Port 22 User root HostName serveur.exemple.com

A partir de maintenant, un simple ssh serveur suffira pour ce connecter sur la machine en question. Le paramètre **Port 22** est facultatif si le serveur fonctionne sur le port par défaut.

Nous pouvons maintenant aller voir plus bas pour un résumé des <u>commandes de base</u> (http://www.antiseches.net/#cdebases) ou pour apprendre à gérer des <u>tunnels ssh</u> (http://www.antiseches.net/#tunnels).

# PuTTY pour Windows

PuTTY est un client ssh et telnet graphique pour systèmes X-Window et pour windows. Il permet de se connecter à un serveur ssh en cliquant sur une icone. On peut également gérer des profiles, créer des tunnels, émuler des touches de fonctions, etc. Bref il est trés complet et trés simple d'utilisation. Un autre de ses avantages par rapport à openssh pour windows est de permettre les copier/coller trés facilement : la copie se fait tout simplement en sélectionnant le texte alors que le collage se fait avec un clic droit sur la souris.

# Téléchargement du logiciel

http://www.chiark.greenend.org.uk/~sgtatham/putty/ (http://www.chiark.greenend.org.uk/%7Esgtatham/putty/) http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html (http://www.chiark.greenend.org.uk/%7Esgtatham/putty/download.html)

# Installation

Démarrer l'installation de PuTTY en double-cliquant sur l'icone, et suivre les étapes suivantes :







PuTTY est maintenant installé.

#### Utilisation de base

Maintenant que PuTTY est installé, on peut commencer à l'utiliser de facon tout à fait basique pour se connecter à une machine distante, soit en telnet soit en ssh. Pour cela, on lance le logiciel et on remplit les champs suivants :

Host Name or IP address : le non FQDN de la machine distante ou son adresse IP. Comme pour openssh, on peut mettre **user@serveur.exemple.com** pour se connecter avec un nom d'utilisateur particulier. Si on ne met pas **user@**, ne nom d'utilisateur sera demandé à chaque connexion.

**Protocol** : Telnet ou SSH. le  $n^{\circ}$  de port par défaut change alors automatiquement : 22 pour SSH et 23 pour telnet.

Saved Session : un nom pour identifier la session si on désire la sauvegarder.

**Close window on exit** : permet de fermer ou non le fenêtre lorsque le connexion est fermée. Généralement on sélectionne **Only on clean exit** ce qui permet de garder les infos à l'écran en cas de coupure réseau.

Si on veut sauvegarder la session, on appuie sur le bouton Save.

Coorden	-	Paula antines for your D.J.T.M.s	a a a la a
- Logging ∃ Terminal - Keyboard - Bell		Specify your connection by host name or Host Name (or IP address) serveur, example, com	IP address Port 22
- Features Window - Appearance - Behaviour - Translation		Protocol:         O Baw         O Ielnet         O Rlogin           Load, save or delete a stored session         Saved Sessions         Saved Sessions	<u>⊙ s</u> sн
Colours Colours Data Proxy Telnet Riogin	a	Default Settings	Load Saye Delete
B-SSH Kex Xith Xiti Tunnels	~	Close window on exit: Always Never Oldrig on	clean exit

## Création d'une paire de clés

Pour éviter de se connecter en étant obligé de saisir le mot de passe à chaques fois, nous allons comme pour openssh créer une paire de clés et les utiliser pour s'authentifier.

Pour créer cette paire de clés, on utilise le programme PuTTYgen qui est fournit avec PuTTY :



Une fois lancé, le programme ressemble à ca :

le <u>K</u> ey Con <u>v</u> ersions <u>H</u> elp		
Key		
No key.		
The management of the contract of the		
Actions	r	
Actions Generate a public/private key pair	[	( <u>G</u> enerate )
Actions Generate a public/private key pair Load an existing private key file	[	( <u>G</u> enerate)
Actions Generate a public/private key pair Load an existing private key file	[ Counciliation ]	( <u>G</u> enerate) Load
Actions Generate a public/private key pair Load an existing private key file Save the generated key	Save public key	Load
Actions Generate a public/private key pair Load an existing private key file Save the generated key Parameters	Save public Key	Load
Actions Generate a public/private key pair Load an existing private key file Save the generated key Parameters Type of key to generate: SSH2 BSJ SSH2 BSJ	Save public key	[ Generate ] Load Seve private key 12 DSA

Aprés avoir sélectionné le type et la longueur de la clé dans **Parameters**, on appuie sur le bouton **Gererate**. Le programme demande alors de faire des mouvements aléatoires avec la souris pour que les clés soient générées :

Key		
Please generate some randomness by mo	oving the mouse over the bla	nk area.
Antinana		
Actions Generate a public/private key pair		Generate
Actions Generate a public/private key pair		Generate
Actions Generate a public/private key pair Load an existing private key file		<u>G</u> enerate Lord
Actions Generate a public/private key pair Load an existing private key file Save the generated key	Seve public key	Generate Loed Save private key
Actions Generate a public/private key pair Load an existing private key file Save the generated key Parameters	Seve public key	Generate Loss Save private key
Actions Generate a public/private key pair Load an existing private key file Save the generated key Parameters Type of key to generate: SOPHICESAT	Seve public key	Genorate Load Save privade key

A la fin, on a la fenêtre suivante :

PuTTY Key Gen	erator		?
<u>Key Conversion</u>	ns <u>H</u> elp		
Key			
Public key for pastir	ng into OpenSSH authoriz	ed_keys file:	
ssh-rsa AAAAB3NzaC1yc2 GMb3GP76ISrf2px bsS1IkiA8uSIC69d rsa-key-20050323	EAAAABJQAAAIEAImYD U+/BP?mywJdRGmait/hi u/8wVLLvOIJL6MQrqesM	Nngp+24G+9zZ083Y1 INTTKZigKM/vz+ZH IKIFJEVUKI4nDJIAYFB	R0FZ0N0Vr4F4fSV vDHW/d+2st5g72K2q YtpK7z79ngiZgM=
Key fingerprint	ssh-rsa 1024 b2:97:fc	19:e4:01:52:08:a4:09	(af:7d:04:3a:1a:a1
Key <u>c</u> omment	Isa-key-20050323		
Key passphrase:			
Confirm passphrase			
Actions			
Generate a public/p	orivate key pair		Generate
Load an existing pri	vate key file		Load
Save the generated	l key	Save pyblic key	Save private key
Parameters			
Type of key to gene	state:	0.5	SH2 DSA
C SSH1 (RSA)	<ul> <li>april 104</li> </ul>		

On entre alors un éventuel commentaire dans le champ **Key Comment** puis une passphrase dans les champs **Key passphrase** et **Confirm passphrase**. Bien qu'elle augmente la sécurité, l'utilisation d'une passphrase complique l'authentification par clés. Nous n'en mettons donc pas.

Ensuite, on sauvegarde les clés sur le disque dur :

PuTTY Key General	lar			9 I VI
le Key Conversions	Save public key	as:		? >
Key	Enregistrer sous :	🔄 keys	- 6	s) 🛃 🔳 🖻
Public key for pasting i				
ssh-rea AAAAB3NzaC1yc2EA GMb3GP76ISrlZpxU+ bcS1lkiA8uSIC69du/8 rsa-key-20050323				
Key fingerprint:				
Key comment				
Key passphrase:	Nom du fichier :	orion-rsa-pub-key		Enregistrer
Confirm passphrase:	Enregistrer gous :	All Files (*.*)	¥	Annuler
Actions		C Ouvrir en lecture seu	le .	
Generate a public/prive			() ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( )	
Load an existing private	e key file		Load	1
Save the generated ke	y	Save p <u>u</u> blic key	Save private key	Ī
Parameters				
Type of key to generate C SSH1 (RSA)	ssh2B	sa oss	H2 <u>D</u> SA	

Sauvegarde de la clé publique

Eanic ki					
ssh-rsa AAAAB: GMb3G bsS1lkir rsa-key- Key fjng Key gar	Enregistrer sgus : [	🚖 keys		•	<u></u>
Key pas: Confirm ( Actions -	Nom du fichier :	orion-rsa-privkey	4		Erregistre
Key pas Confirm ( Actions Generation	Nom du fichier : Enregistrer gous :	orion-rsa-privkey PuTTY Private	i Key Files (".ppk.)	•	<u>E</u> nregistre Annuler
Key pas Confirm ( Actions Generational Load an	Nom du fichier : Enregistrer gous :	orion-rsa-privkey PuTTY Private	( Key Files (*.ppk) ture seule	•	<u>E</u> nregistre Annuler
Key pass Confirm ( Actions - Generation Load an Save the	Nom du fichier : Erregistrer gous : generated key	orion-rea-privkey PuTTY Private	( Key Files (*,ppk) ture seule Save pyblic key	▼ Save p	<u>E</u> nregistre Annuler rivate key

Sauvegarde de la clé privée

Une fois les clés sauvegardées, il faut copier le contenu de la clé publique sur le serveur distant. Attention, avec PuTTY on ne copie pas le fichier en entier, mais on fait un copier/coller de la chaine de caractères qui se trouve dans le champ "Public key for pasting into OpenSSH authorized\_keys file" :

Ele	Key	Conversions	Help	
K	eu			
4	Ublic I	key for pasting i	into OpenSSH authorized_keys file>	
	sh-isa AAAAE	3NzaC1yc2EA	AAABJQAAAIEArlAS+DCoRnkDOui6irhUiay8%	As

Pour copier le contenu dans le fichier en question, il y a plusieurs méthodes, mais la plus simple est de :

- Se connecter au serveur en ssh avec PuTTY.
- Editer le fichier ~/.ssh/authorized\_keys par exemple avec vi.
- Copier la chaine de caractères à la fin du fichier en faisant un clic droit sur la souris.

Configuration de PuTTY pour se connecter sans mot de passe

Maintenant que la clé est copiée sur le serveur distant, il faut configurer la session dans PuTTY en indiquant le chemin vers la clé privée que l'on vient de créer et qui est stockée sur le disque local.

On charge donc la session dans PuTTY en la sélectionnant dans **Saved Sessions** et en cliquant sur le bouton **Load.** Ensuite on va dans la fenêtre de gauche, on déroule l'arborescence pour aller dans **Connection / SSH / Auth** puis dans la case **Private key for file Authentication** et on indique le chemin de la clé privée :

ategory:		
Session     Logging     Terminal     Keyboard     Bel     Features     Window     Appearance     Behaviour     Translation     Selection     Colours	п ()	Options controlling SSH authentication           Authentication methods           Attempt TIS or ClyptoCard auth (SSH-1)           Attempt TIS or ClyptoCard auth (SSH-2)           Authentication parameters           Allow agent (orwarding           Allow attempted changes of genrame in SSH-2           Private key life for authentication:           D:\cle priv ppk
Connection Data Proxy Tehet Rilogin SSH Kex Auth -X11 Turnets		

On revient ensuite dans la fenêtre principale et on sauvegarde la session avec le bouton **Save**. On peut enfin se connecter sans utiliser de mots de passe. Il ne faut pas oublier de mettre le nom du user lorsque on renseigne l'adresse IP ou le nom du serveur :

egoty.	-		1
Session	^	Basic options for your PuTTY se	ssion
Logging		- Specify your connection by host name or	P address
Terminal		Host Name (or IP address)	Port
- Keyboard		(root@serveut.exemple.com	22
- Bel		Detred	
- Features		Raw Telest OBlogin	(Laso
Appendix		OTow OTener Oundin	0 2011
Rehaviour		- Load, save or delete a stored session	
Translation		Saved Sessions	
Selection		serveur.exemple.com	1
Colours		D. L. N.C. W.	
Connection		Server a everyole com	Load
- Data			Caus
Proxy			2.47.6
Telnet			Delete
Blogin		×	
B-SSH			
Kex		Charles and the second	
Auth		Close window on ext:	fue as a
-X11		Chanaps Chevel Clouverte	(collicit)
Termola	20		

Le paramètrage est mainenant terminé, il suffit de faire un double-clic sur le profile pour pouvoir se connecter.

On maintenant utiliser PuTTY pour se connecter rapidement sur un serveur distant. Si on veut se connecter encore plus rapidement, il suffit de créer un raccourci vers l'icone de l'éxécutable et de modifier la ligne "cible" du raccourci en y incluant -load nom\_du\_profile. Par exemple si on a paramétré un profile qui s'appelle serveur.exemple.com, il suffit de créer un raccourci sur le bureau windows en mettant ceci sur la ligne "cible" :

"C:\Program Files\PuTTY\putty.exe" -load serveur.exemple.com

Propriétés de l	PuTTY	? 🔀
Général Racco	urci Compatibilité Sécurité	
🛃 Р	TTY	
Type de cible :	Application	
Emplacement :	PuTTY	
<u>C</u> ible :	Files\PuTTY\putty.exe" -load server.exemple.com	
Démarrer dans :		
Touche de raccourci :	Aucun	
Exécuter :	Fenêtre normale	
Comm <u>e</u> ntaire :	SSH, Telnet and Riogin client	
Rechercher la	cible Changer d'icône Ayancé	
	OK Annuler Ap	pliquer

Ensuite il n'y a plus qu'à faire un double-clic sur ce raccourci pour se retrouver connecté à la machine **serveur.exemple.com**.

#### PuTTYtray : une version améliorée de PuTTY

Comme le code de PuTTY est librement accessible, on peut trouver plusieurs variantes de ce logiciel sur Internet, dont <u>PuTTYtray (http://haanstra.eu/putty/)</u> qui possède des améliorations intéressantes :

- Possibilité de se mettre dans le systray quand on réduit la fenêtre.
- Peut reconnecter une session quand un ordinateur sort de veille.
- Permet de rendre les hyperliens cliquables.

- ...

Pour bénéficier de toutes ces améliorations, il suffit de télécharger le binaire putty.exe et de remplacer l'original dans le répertoire d'installation. On peut différencier les 2 binaires putty.exe par leur icone qui est différente.

Dans le même style il y a également <u>Kitty (http://www.9bis.net/kitty/)</u> qui semble intéressant mais que je n'ai pas testé. Kitty permet d'intégrer des fonctions de transfert de fichier grâce à pcsp.exe.

#### Putty Connexion Manager

Putty Connexion Manager (http://puttycm.free.fr) est un logiciel gratuit qui permet de gérer les connexions PuTTY et d'ouvrir plusieurs connexions sous forme d'onglets dans le même fenêtre :

Ele Database gen	v look ( woord 😅 🖬 😋 🖉 🏈	Decornect Configuration a Put TY System menu		1912
9 Host	Protocol Tellmen	- Sesson Default Settings - Logo	Password	0
/ Ny Database serve	u u	• X Poder*	* × =	Provide an original service and a servi

+1	1
Nywebiawar + X	
	-
Sun-28 5.8	
logini [	
	1
Connection Router spaced	AI

Cela peut s'avérer très pratique à l'utilisation.

# Création de tunnels sécurisés

De nos jours internet est tellement peu sûr qu'il faut trouver des moyens pour sécuriser les communications, surtout si des informations doivent circuler en clair. Pour sécuriser ces communications, une technique simple est d'utiliser un serveur ssh et de créer des tunnels ssh entre le client et les differents services et/ou serveurs distants.

Il y a deux méthodes pour créer un tunnel ssh en ligne de commande. La première, qui est aussi la plus utiliée consiste à créer le tunnel à partir du client, tandis que dans la deuxième permet de créer le tunnel à partir du serveur. Nous ne nous attarderons pas sur la deuxième car elle trés peu utilisée.

Passons maintenant à la pratique.

#### Tunnel ssh simple

Commençons par un exemple simple. Dans ce cas, on veut simplement sécuriser les échanges de données entre un client et un serveur http sur lequel tourne également un serveur ssh. Bien sûr on pourrait utiliser https à la place, mais ce n'est pas le but. Dans cet exemple on utilise seulement 2 machines, le client et le serveur :



Pour créer le tunnel ssh à partir de la machine cliente (Windows, Linux ou MacOSX) ouvrir un shell et entrer la commande suivante :

```
ssh -L 200:serveur1:80 serveur1
ou
ssh -L 200:serveur1:80 user@serveur1
ou encore
ssh -l user -L 200:serveur1:80 serveur1
```

Explication de la syntaxe : ssh -L portlocal:serveur1:portdistant serveur1

-L signifie que c'est une redirection de port locale, c'est à dire initiée par le client portlocal est le n° de port tcp qu'il faudra utiliser pour se connecter sur localhost. Ce n° de port est arbitraire, mais il faudra néanmoins être root sur la machine client s'il est inferieur à 1024.

portdistant est le port utilisé par le service du serveur distant, donc 80 pour un serveur http.

serveur1 est le nom de la machine sur lequel tournent les serveurs http et ssh. A la place des noms des serveurs, on peut aussi utiliser les adresses IP.

-1 user de la 3° commande est l'équivalent de **user**@ de la 2° commande. Dans la première forme de syntaxe, on ne spécifie pas le nom de **user** car on part du principe que c'est le même que sur le client.

Une fois le tunnel créé, il suffira de taper http://localhost:200 dans la barre d'adresse du navigateur pour être en fait connecté au serveur distant. On obtiendra alors les mêmes informations que si on avait taper http://serveur1, mais les échanges de données ne pourront pas être écoutées.



# Tunnel ssh en utilisant une passerelle pour atteindre une autre machine

Dans ce cas de figure, on utilise un serveur ssh comme passerelle pour joindre une autre machine. Les échanges cryptés ne se font alors que entre le client (**hote1**) et la passerelle (**serveur1**), et c'est cette dernière qui relaye les informations en clair à la machine que l'on souhaite atteindre. Cette topologie peut être utilisée pour résoudre les problèmes suivants :

- cas n° 1: on souhaite atteindre une machine (**le serveur2**) dans un réseau privé, mais cette machine n'a pas de serveur ssh. Les informations qui transitent sur internet doivent être cryptées, mais pas les informations qui circulent sur le réseau local. Exemple : on est connecté à un hotspot wifi public et on veut utiliser le serveur intranet de son entreprise.

- cas n° 2 : on est sur un réseau local et on désire se connecter à un serveur de news (nntp) qui est le serveur 3. Malheureusement le firewall du réseau local interdit ce genre de connexions. Par contre, on a la possibilité de se connecter sur un serveur ssh à l'extérieur et de s'en servir de passerelle pour atteindre ce serveur de news. Bien sûr le firewall du serveur doit laisser passer les connexions sortantes sur le port tcp 22 pour que ca marche.

Notons que dans ces 2 cas de figures, le serveur ssh est lui aussi derrière un routeur et a une adresse IP privée. Le routeur distant devra donc faire du Port Forwarding, c'est à dire qu'il faut le configurer pour que lorsqu'il recoive une demande de connexion sur le port TCP 22 (ssh) de son adresse IP publique, il forwarde la requete vers le port TCP 22 de l'adresse IP privée du serveur ssh.



Que l'on souhaite joindre le **serveur2** ou le **serveur3**, la commande sera la même. A partir de la machine cliente (Windows, Linux ou MacOSX) ouvrir un shell et entrer la commande suivante :

```
ssh -L 200:serveur2:119 serveur1
ou
ssh -L 200:serveur2:119 user@serveur1
ou encore
ssh -l user -L 200:serveur2:119 serveur1
```

On voit que la syntaxe est quasiment la même que dans le cas précédant. On remplace juste le serveur de destination par **serveur2** au lieu de **serveur1**. Une fois le tunnel créé, il suffira de parametrer le client nntp pour qu'il se connecte sur le port 200 de **localhost**. Voiçi un exemple de configuration avec thunderbird :

Nouvelles et Blogs	Paramètres du serveur
Espace disque	Type de serveur : Serveur de forums (NNTP)
News	Nom du serveur : localhost Port : 200 Défaut : 119
Paramètres serveur     Conies et dossiers	Paramètres de sécurité
Rédaction et adressage	Utiliser une connexion sécurisée (SSL)
-Hors ligne et espace disque	Paramètres du serveur
Paramètres serveur	Vérifier les nouveaux messages toutes les 10 minutes.
-Copies et dossiers Rédection et adressage	Me demander avant de télécharger plus de 500 messages.
Hors ligne et espace disque	Toujours demander une authentification lors de la connexion à ce serveur

# Tunnels SSH avec PuTTY

PuTTY permet également de créer des tunnels ssh. En fait il fait la même chose que les commandes précédentes, mais il suffit de cliquer sur un bouton pour monter le tunnel.

Pour monter un tunnel simple (cf notre premier cas), il faut configurer une session comme on vient de le voir précédemment. Ensuite, dans le menu de gauche, dérouller l'arborescence des menus PuTTY pour aller **Connections / SSH / Tunnels**. On renseigne alors les champs **Source Port** et **Destination** de la façon suivante :

🞇 PuTTY Configuration	2 🛛
Category: Logging  Terminal Category: Categor	Options controlling SSH poit forwarding         Poit forwarding         Local ports accept connections from other hosts         Remote gorts do the same (SSH-2 only)         Forwarded ports:         Remove         Add new forwarded port:         Source port       200         Add         Destination         serveur1:80         Source         Auto
-X11 Tunnels Bugs	

Ensuite on appuie sur le bouton Add et on a l'écran suivant :

😹 PuTTY Configura	rtion	2 🔀
Category:		
- Logging	^	Options controlling SSH port forwarding
Terminal		Port forwarding
Keyboard		Local ports accept connections from other hosts
Bel		Bemote ports do the same (SSH-2 only)
Features		Forwarded ports:
Appearance		<u>H</u> emove
Behaviour		L200 serveur1:80
- Translation		
Selection		
- Colours		Add new forwarded port:
Connection	=	Source port Add
Data		Destination
Tehet		Oleand ORemote ODunamic
Blogin		Auto DIPut DIPut
E-SSH		
Kex		
Auth		
X11		
- Tunnels		
Bugs	<b>M</b>	
About	<u>H</u> elp	<u>Open</u>

On revient dans **Session** sur la fenêtre de gauche et on appuie sur le bouton **Save** de la fenêtre de droite. On peut maintenant se connecter comme si on avant entré la commande :

ssh -l root -L 200:serveur1:80 serveur1.

Si on veut utiliser PuTTY pour créer un tunnel pour se connecter sur une machine tiers (cf notre cas n° 2), on procéde exactement de la même façon mais à la place de serveur1:80 on écrit serveur2:80 ou serveur3:80 dans le champ Destination du formulaire de PuTTY. Par contre, on ne change rien dans le champ Host name or IP address du menu Session. On a alors l'équivalent de la commande :

```
ssh -l root -L 200:serveur2:80 serveur1
```

Proxy Socks5 avec Putty

Putty peut être utilisé comme client **Proxy Sock 5**. La configuration de cette fonctionnalité est identique à celle d'un tunnel avec les paramètres suivants (par exemple) :

Source Port : Entrer un port local qui peut être utilisé sur le PC (par exemple 8080) Destination : Laisser le champ vide et sélectionner l'option Dynamic à la place de Local

alegoly.				( F
<ul> <li>Session</li> <li>Logging</li> <li>Terminal</li> <li>Keyboard</li> <li>Bell</li> <li>Features</li> <li>Window</li> <li>Appearance</li> <li>Behaviour</li> <li>Translation</li> <li>Selection</li> <li>Colours</li> </ul>	•	Option Port forwarding Local ports Remote port Forwarded port Add new forwar Source port	is controlling SSH port accept connections fir its do the same (SSH-2 is: inded port: 1 8080	forwarding om other hosts conly) <u>R</u> emove
<ul> <li>Connection</li> <li>Data</li> <li>Proxy</li> <li>Telnet</li> <li>Rlogin</li> <li>SSH</li> <li>Kex</li> <li>Auth</li> <li>TTY</li> <li>X11</li> </ul>	2	Destination	O Remote	O Dynamic O IPv <u>6</u>

Cliquer ensuite sur le bouton Add et vérifier qu'on obtienne bien l'écran suivant :

Category:  Category:  Session  Cogging  Forwarding  Connection  Colours  Category:  Depring  Depring	🔀 PuTTY Configur	ation			? 🛛
Session     Options controlling SSH port forwarding     Options	Category:	_			
CUTIFIC CUUF	<ul> <li>Session         <ul> <li>Logging</li> <li>Terminal</li> <li>Keyboard</li> <li>Bell</li> <li>Features</li> </ul> </li> <li>Window         <ul> <li>Appearance</li> <li>Behaviour</li> <li>Translation</li> <li>Selection</li> <li>Colours</li> </ul> </li> </ul>		Option Port forwarding Local ports Remote port Forwarded ports D8080 Add new forward Source port	s controlling SSH p accept connection is do the same (SS s: ided port:	s from other hosts H-2 only) <u>R</u> emove
Data Proxy Telnet Riogin SSH Kex Auth TTY -X11 ▼	Data - Data - Proxy - Telnet - Rlogin - SSH - Kex - Auth - TTY - X11	~	Destination <u>L</u> ocal Auto	O Remote O IPv <u>4</u>	⊙ D⊻namic O IPv§

Sauvegarder la session Putty et lancer la connexion.

Exemple d'utilisation : surfer de façon sécurisée avec Firefox

Nous allons paramètrer firefox pour que le traffic qu'il engendre soit chiffré et donc qu'il ne puisse pas être intercepté par un sniffer réseau ou un firewall. Cette configuration se fait en 2 étapes : le traffic http et les requêtes DNS.

#### Configuration du traffic http

Aller dans le menu Outils / Options, puis onglet Réseau et enfin bouton Paramètres :

néral Réseau	Mises à jour	Chiffrement	vie privee	Securice	Avalice	
Connexion Configurer la fa	;on dont Fire	fox se connecte	à Internet		Ea	ramètres
Stockage hors o	onnexion 50 💽 1	Mo d'espace disqu	e pour le cache		jider le cache r	naintenant
utilisation h Les sites suivan connexion :	ors connexic ts ont des de	in onnées enregistré	ies pour une ut	ilisation hors	[E2	ceptions
						unnalmer
						uppriner

Renseigner les champs suivants :

- Configuration manuelle du Proxy

- Hôte SOCKS : 127.0.0.1 et 8080 pour être en phase avec la configuration de notre exemple ci-dessus

#### - SOCKS V5

Configuration du serveur ; Pas de proxy Détection automatiqu Utiliser les paramètres	proxy pour accéder à Internet e des paramètres de proxy pou s prox <u>y</u> du système	r ce réseau	
<ul> <li>Configuration manual</li> </ul>	le du proxy :		>
Proxy HTTP :		Port :	0 🗢
	Ubliser ce serveur proxy p	our tous les protocoles	
Proxy <u>S</u> SL :		Port :	0 🗘
Proxy ETP :		Pogt :	0 🗢
Proxy gopher :		Port :	0
mote SOQKS :	127.0.0.1	Por <u>t</u> :	8080 🧅
	O SOCKS v4 ⊙ SOCKS v	5	and the second
Pas de pro <u>x</u> y pour :	localhost, 127.0.0.1, 172.16.	204.0/24	
	Exemples : .mozilla.org, .asso.	fr, 192.168.1.0/24	



Puis valider en cliquant sur OK et fermer la fenêtre des options.

#### Configuration des requetes DNS

Si on ne passe pas par cette deuxième étape, le traffic http sera crypté mais les requêtes DNS continueront de se faire en clair par l'intermédiaire du serveur DNS qui se trouve sur le réseau local.

Aller dans la barre d'adresse de firefox et taper about:config

Taper dns dans la zone de filtre

Localiser la chaine **network.proxy.socks\_remote\_dns** et la modifier pour qu'elle soit à **True.** Si elle n'existe pas, il suffit de la créer.

				-
	about:config	🔹 🖓 🗙 🛐	· • •	<b>AB</b>
•	Q			
🗧 😢 🔹 🐼 ab 🗶 😕	۵ - 🖞 - 🕲	• 📥 🖶 • 🛛	<b>}• ⊗•</b> (	<u>۲</u>
iltre : dns				×
Nom de l'option	/ Statut	Туре	Valeur	
network.dns.disableIPv6	par défaut	booléen	false	
etwork.dns.ipv4OnlyDomains	par défaut	chaine		
etwork.proxy.socks_remote	_dns défini par l'utilisati	eur booléen	true	

Redémarrer Firefox. Celui-ci est maintant prêt à être utilisé pour surfer en utilisant le serveur ssh distant comme proxy. Toutes les communications entre firefox et le serveur ssh seront cryptées.

#### Gestion des tunnels ssh sous Windows

Parfois, on veut juste créer un tunnel ssh mais on ne veut pas ouvrir de terminal.

Il existe plusieurs logiciels qui permettent de faire ca. Le logiciel reste ouvert en tache de fond et se met dans le Systray à coté de l'horloge. Parmis ces logiciels, on peut citer :

- XShell (http://www.netsarang.com/products/xsh%5Ffeatures.html) (payant)
- PuTTYtray (http://haanstra.eu/putty/) (gratuit)
- Tunnelier (http://www.bitvise.com/tunnelier) (gratuit pour une utilisation non commerciale)
- MyEnTunnel (http://nemesis2.qx.net/pages/MyEnTunnel) (gratuit)
- PuTTY Tunnel Manager (http://code.google.com/p/putty-tunnel-manager/) (gratuit)

Mon choix se porte sur le dernier, PuTTY Tunnel Manager car il s'agit d'un simple exécutable. Il suffit de le télécharger dans le même répertoire que PuTTY et de l'éxécuter.

Quand on le lance, il propose les profils déjà configurés avec PuTTY, puis il se met dans la barre du Systray et tourne en tache de fond.

# Résumé des commandes de base

Connexion simple à un serveur : ssh user@serveurdistant OU ssh -1 user serveurdistant

Connexion en utilisant un autre port que celui par défaut ssh -p 2222 user@serveurdistant OU ssh -p 2222 -1 user serveurdistant

Connexion à un serveur et exécution de commandes sur celui-ci ssh -l user serveurdistant "uptime"

Transfert de fichier en utilisant la copie sécurisée scp ou sftp

scp /tmp/fichier user@serveurdistant:/tmp
scp user@serveurdistant:/tmp/fichier /tmp
sftp -o Port=22 user@serveurdistant:/tmp/fichier /tmp/

#### Création de tunnels ssh

ssh -l user -L portlocal:serveurcible:portcible serveurssh

Rémi TAUPIN, 2011 - Propulsé par Gandi Hébergement avec DotClear 2 - Informations légales