Connexion avec « ssh » sans mot de passe [1]

Version :V1, date création : 15/01/2008.

1 Qu'est-ce que SSH ?

SSH signifie *Secure SHell*. C'est un protocole qui permet de faire des connexions sécurisées (i.e. cryptées) entre un serveur et un client SSH.
SFR utilise le programme OpenSSH, la version libre du client et du serveur SSH.

2 Différences entre « telnet » et « ssh »

La connexion à distance à un serveur, par « ssh » est plus sécurisée que la connexion par « telnet ». Telnet est un protocole permettant d'accéder à distance à une machine, mais la connexion n'est pas sécurisée : le mot de passe et les données sont transférés en clair. Telnet ne permet pas de faire des transferts de fichiers. Il est donc conseillé de ne pas utiliser Telnet, mais uniquement ssh.

[1] Ce document a été réalisé à partir des informations contenues dans ces 3 documents :
a) clé ssh : http://lipforge.ens-lyon.fr/www/webarenaire/cle-ssh.html
b) Partie IV. Debian GNU/Linux en réseau. Chapitre 1. L'accès à distance par SSH : http://formation-debian.via.ecp.fr/ch35.html
c) ssh without password : http://unix.ittoolbox.com/groups/technical-functional/solaris-l/ssh-without-password-765537

3 Cryptographie asymétrique

« ssh » utilise la cryptographie asymétrique RSA ou DSA. En cryptographie asymétrique, **chaque personne dispose d'un couple de clé : une clé publique et une clé privée**.
La clé publique peut être librement publiée tandis que la clé privée doit rester secrète. La connaissance de la clé publique ne permet pas d'en déduire la clé privée.
Si la personne A veut envoyer un message confidentiel à la personne B, A crypte le message avec la clé publique de B et l'envoie à B sur un canal qui n'est pas forcément sécurisé. Seul B pourra décrypter le message en utilisant sa clé privée.

4 Cryptographie symétrique

« ssh » utilise également la cryptographie symétrique.
Son principe est simple : si A veut envoyer un message confidentiel à B, **A et B doivent d'abord posséder une même clé secrète**. A crypte le message avec la clé secrète et l'envoie à B sur un canal qui n'est pas forcément sécurisé. B décrypte le message grâce à la clé secrète. Toute autre personne en possession de la clé secrète peut décrypter le message.

La cryptographie symétrique est beaucoup moins gourmande en ressources processeur que la cryptographie asymétrique... mais le gros problème est l'échange de la clé secrète entre A et B.
Dans le protocole SSL, qui est utilisé par SSH et par les navigateurs Web, **la cryptographie asymétrique est utilisée au début de la communication pour que A et B puissent s'échanger une clé secrète de manière sécurisée**... puis la suite la communication est sécurisée grâce à la cryptographie symétrique **en utilisant la clé secrète échangée**[2].

5 Mécanisme d'établissement d'une connexion SSH

Un serveur SSH dispose d'un couple de clés *RSA* stocké dans le répertoire « **./etc** » [3] [4] et généré lors de l'installation du serveur.
Le fichier **ssh_host_rsa_key** contient la clé privée et a les permissions **600**.
Le fichier **ssh_host_rsa_key.pub** contient la clé publique et a les permissions **644**.

Nous allons suivre par étapes l'établissement d'une connexion SSH :
1. Le serveur envoie sa clé publique au client.

[2] Pour plus d'informations sur la cryptographie, le dossier consacré à ce sujet par le magazine pour la science dans son hors-série de Juill.et-Octobre 2002.
[3] Sur Sun, ce répertoire pourra être « /opt/CSIossh/etc » ou « /usr/local/etc ». (Le chemin de ce répertoire « etc » dépend du chemin d'installation du package Openssh).
[4] Exemple du contenu d'un répertoire « etc » (ici le répertoire « /usr/local/etc » d'un serveur AIX) :

```
.   banner.ssh.1 ssh_config.default  ssh_host_key.pub       ssh_prng_cmds.default
..  moduli ssh_host_dsa_key          ssh_host_rsa_key       sshd_config
banner.bts ssh_banner_message_fr ssh_host_dsa_key.pub ssh_host_rsa_key.pub sshd_config.default
banner.ssh ssh_config            ssh_host_key         ssh_prng_cmds        sshd_config.orig
```

2. Le client génère une clé secrète et l'envoie au serveur, en cryptant l'échange avec la clé publique du serveur (cryptographique asymétrique). Le serveur décrypte la clé secrète en utilisant sa clé privée, ce qui prouve qu'il est bien le vrai serveur.
3. Pour le prouver au client, il crypte un message standard avec la clé secrète et l'envoie au client. Si le client retrouve le message standard en utilisant la clé secrète, il a la preuve que le serveur est bien le vrai serveur.
4. Une fois la clé secrète échangée, le client et le serveur peuvent alors établir un canal sécurisé grâce à la clé secrète commune (cryptographie symétrique).
5. Une fois que le canal sécurisé est en place, le client va pouvoir envoyer au serveur le login et le mot de passe de l'utilisateur pour vérification. Le canal sécurisé reste en place jusqu'à ce que l'utilisateur se déloggue.

La seule contrainte est de s'assurer que la clé publique présentée par le serveur est bien sa clé 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 par exemple de demander à l'administrateur du serveur quelle est le « *fingerprint* » [5] de la clé publique du serveur avant de s'y connecter pour la première fois.
Le « *fingerprint* » d'une clé publique est une chaîne de 32 caractères hexadécimaux unique pour chaque clé [6], servant de mécanisme d'authentification.
Il s'obtient grâce à la commande « **ssh-keygen** » (voir plus loin).

6 Installation et configuration de SSH

6.1 Installation du client et du serveur SSH

Sur la plupart des OS, le client SSH est préinstallé.
Pour pouvoir se connecter à distance, il faut l'installer le serveur SSH [7], par une commande comme :
??? (à compléter)
L'installation comporte une étape de génération des clefs de cryptage.
Finalement, le serveur SSH se lance.

6.2 Configuration du serveur SSH

Le fichier de configuration du serveur SSH est « sshd_config » (qu'on trouve dans le répertoire « etc » précédent).
Il ne faut pas confondre ce fichier avec le fichier « ssh_config » (situé aussi dans le répertoire « etc » précédent), qui est le fichier de configuration du client SSH.

[5] « *fingerprint* » signifie en français « empreinte digitale ».
[6] Exemples, de « fingerprint » : 0a:e7:c2:5d:fe:4f:7b:25:65:cc:f8:54:d9:d9:7a:5e , b2:05:ed:62:11:4d:5d:1d:d0:82:01:6a:55:26:99:af , c8:12:8a:9e:b9:30:b9:9c:86:8a:68:09:cb:f4:24:73 etc.
[7] Sur BSD, le client SSH est disponible dans le paquet openssh-client, et la commande d'installation est :
apt-get install openssh-server

Exemple de fichier « sshd_config » :

```
AuthorizedKeysFile /usr/local/etc/authorized_keys/%u
Banner /usr/local/etc/banner.ssh
ChallengeResponseAuthentication no
Ciphers aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes128-ctr
HostbasedAuthentication no
HostKey /usr/local/etc/ssh_host_dsa_key
IgnoreRhosts yes
IgnoreUserKnownHosts no
LoginGraceTime 60
LogLevel VERBOSE
MACs hmac-md5,hmac-sha1,hmac-ripemd160,hmac-sha1-96,hmac-md5-96
MaxStartups 10
PasswordAuthentication yes
PermitEmptyPasswords no
PermitRootLogin yes
PermitUserEnvironment no
PidFile /var/run/ssh.pid
Port 22
PrintLastLog yes
PrintMotd yes
Protocol 2
PubkeyAuthentication yes
RhostsRSAAuthentication no
RSAAuthentication no
StrictModes yes
Subsystem sftp /usr/local/libexec/sftp-server
SyslogFacility AUTH
TCPKeepAlive yes
UseLogin no
UsePrivilegeSeparation yes
X11Forwarding yes
```

=> Commentaire sur les lignes les plus importantes de ce fichier de configuration :

Port 22
Signifie que le serveur SSH écoute sur le port 22, qui est le port par défaut de SSH. Vous pouvez le faire écouter sur un autre port en changeant cette ligne. Vous pouvez aussi le faire écouter sur plusieurs ports à la fois en rajoutant des lignes similaires.

PermitRootLogin yes
Signifie que vous pouvez vous logguer en root par SSH. Vous pouvez changer et mettre "*no*", ce qui signifie que pour vous connecter en root à distance, vous devrez d'abord vous connecter par SSH en tant que simple utilisateur, puis utiliser la commande **su** pour devenir root.
Sans cela, un pirate n'aurait qu'à trouver le mot de passe du compte root, alors que là, il doit trouver votre login et votre mot de passe.

X11Forwarding yes
Signifie que vous allez pouvoir travailler en export display [8] par SSH.

Si vous avez modifié le fichier de configuration du serveur, il faut lui dire de relire son fichier de configuration :

`# /etc/init.d/ssh reload` (ou bien : **/etc/init.d/sshd stop ; /etc/init.d/sshd start**).

[8] sur la commande « export DISPLAY=@IP:0.0 », voir formation Faire de l'export display du site Internet « Formation debian » : http://formation-debian.via.ecp.fr/ch36.html

Notes : Pour arrêter le demon "**sshd**" : « **/etc/init.d/sshd stop** ».
Pour relancer le démon « sshd » : « **/etc/init.d/sshd start** ».
Si ce démon est lancé, le résultat de la commande « **ps -ef|grep sshd** » donnera par exemple :

```
    root   958     1  0   Nov 06 ?        0:01 /opt/CSIossh/sbin/sshd
```

ou bien :

```
    root  2630     1   0   Nov 06 ?           0:03 /usr/local/sbin/sshd
```

(… suivant le répertoire où a été installé Openssh).

7 Se logguer par SSH

7.1 Authentification par mot de passe

Chaque fois qu'à partir d'un serveur client SSH, on se connecte à un serveur distant SSH, on doit à chaque fois saisir un mot de passe (d'authentification de son login sur le serveur distant). C'est la méthode la plus simple. Pour cela, depuis la machine cliente, tapez :

% **ssh login@***nom_dns_serveur_SSH* (ou bien: **ssh login@***nom_serveur_SSH*).

Exemple : **ssh metrica@barbadine**

Si c'est la première connexion SSH depuis ce client vers ce serveur, il vous demande si le « fingerprint » de la clé publique présentée par le serveur est bien le bon, comme ci-après :

```
% ssh metrica@mache
The authenticity of host 'mache (164.17.73.42)' can't be established.
DSA key fingerprint is c8:12:8a:9e:b9:30:b9:9c:86:8a:68:09:cb:f4:24:73.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'mache,164.17.73.42' (DSA) to the list of known hosts.

SFR

metrica@mache's password:
Sun Microsystems Inc.   SunOS 5.6       Generic August 1997
You have mail.
%
```

Pour être sûr que vous vous connectez au bon serveur, vous devez connaître de façon certaine le « fingerprint » de sa clé publique et la comparer à celle qu'il vous affiche. Si les deux « fingerprints » sont identiques, répondez yes, et la clé publique du serveur est alors rajoutée au fichier « ~/.ssh/**known_hosts** » (si ce fichier n'existe pas encore, il est alors créé).

Si vous vous êtes déjà connecté depuis ce client vers le serveur, sa clé publique est déjà dans le fichier « ~/.ssh/**known_hosts** » et il ne vous demande donc rien.

Ensuite, entrez votre mot de passe... et vous verrez apparaître le prompt du shell (du serveur ssh), comme si vous vous êtiez loggué, en local, sur la machine.

7.2 Authentification par clé

On est souvent confronté au besoin suivant :
Besoin : *A partir d'un serveur client, on peut avoir le besoin de se connecter (loguer) à un serveur, sans devoir saisir son mot de passe, à chaque fois.*

Au lieu de s'authentifier par mot de passe, les utilisateurs peuvent s'authentifier grâce à la cryptographie asymétrique et son couple de clés privée/publique, comme le fait le serveur SSH auprès du client SSH.
Ce mécanisme permet alors de se connecter à répétition au serveur SSH (à partir d'un serveur client SSH), sans devoir saisir un mot de passe à chaque fois.

7.3 Générer ses clés

Pour générer un couple de clés *DSA*, tapez :

% **ssh-keygen -t dsa**

Au lancement de la commande précédente, vous aurez un dialogue comme ci-après :

```
Generating public/private dsa key pair.
Enter file in which to save the key (/users/metrica/.ssh/id_dsa):  <= taper entrée
Enter passphrase (empty for no passphrase):                       <= taper entrée
Enter same passphrase again:                                      <= taper entrée
Your identification has been saved in /users/metrica/.ssh/id_dsa.
Your public key has been saved in /users/metrica/.ssh/id_dsa.pub.
The key fingerprint is:
f1:3f:42:e1:59:0d:18:ee:1e:2a:f2:b3:3e:f3:cb:2c metrica@barbadine
```

Tapez simplement sur la touche « entrée » quand on vous demande un mot de passe.

Les messages du « **ssh-keygen** » ci-avant vous annoncent que deux clés ont été générées :
id_dsa qui est la *clé privée* (et qui reste sur le serveur client SSH)
id_dsa.pub qui est la *clé publique* (et qui normalement sera recopiée sur le serveur distant SSH, afin de permettre une connexion « **ssh** » entre un serveur client SSH et un serveur distant SSH, … sans avoir à saisir de mot de passe)

Notes : a) Les clés *DSA* générées ont par défaut une longueur de 1024 bits, ce qui est aujourd'hui considéré comme suffisant pour une bonne protection. Les clés RSA sont plus sûres que les clés DSA.
b) Avec les anciennes versions de SSH, les clés seront stockées dans « **~/.ssh/identity** » et « **~/.ssh/identity.pub** » ; avec les nouvelles, elles seront stockées dans « **~/.ssh/id_rsa** » et « **~/.ssh/id_rsa.pub** » ou dans dans « **~/.ssh/id_dsa** » et « **~/.ssh/id_dsa.pub** ».

Par défaut (la commande ci-avant « **ssh-keygen** » demande confirmation lors du processus de création), la clé privée est stockée dans le fichier « **~/.ssh/id_dsa** » avec les permissions **600** et la clé publique est stockée dans le fichier « **~/.ssh/id_dsa.pub** » avec les permissions **644**.

Lors de la création, la commande ci-avant « **ssh-keygen »** vous demande une « pass phrase » qui est un mot de passe pour protéger la clé privée. Cette « pass phrase » sert à crypter la clé privée. La « pass phrase » vous sera alors demandée à chaque utilisation de la clé privée, c'est à dire à chaque fois que vous vous logguerez en utilisant cette méthode d'authentification.

Note : Un mécanisme appelé « **ssh-agent** » permet de ne pas rentrer le mot de passe à chaque fois... comme nous le verrons un peu plus loin dans ce chapitre.

Notes :
a) Vous pouvez saisir une « `passphrase` » vide.
b) Vous pouvez à tout moment changer la « pass phrase » qui protège votre clé privée avec la commande « **ssh-keygen –p** ».

7.4 Autoriser votre clé publique

Pour cela, il suffit de copier votre clé publique (du serveur client SSH) dans le (à la fin du) fichier « **~/.ssh/authorized_keys** » du serveur distant SSH, sur laquelle vous voulez vous logguer à distance (voir le dessin Fig. 1 page suivante).

Etant logué avec votre login « login », les commandes suivantes permettent de recopier de la *clé publique* (**id_dsa.pub**) située dans le répertoire « **$HOME/.ssh** » du serveur client SSH, à la fin du fichier « **~/.ssh/authorized_keys** » (situé dans le répertoire « **$HOME/.ssh** » du serveur distant SSH) [9] :

1) % **scp id_dsa.pub** login**@***nom_serveur_distant_SSH***:.ssh/**cle.tmp
Il faut alors rentrer votre mot de passe de votre compte « login » sur serveur distant SSH « *nom_serveur_distant_SSH* ».

Exemple : % **scp id_dsa.pub** *metrica@barbadine***:.ssh/***cle.tmp*

2) se "loguer" sur « *nom_serveur_distant_SSH* », avec votre login « login » :

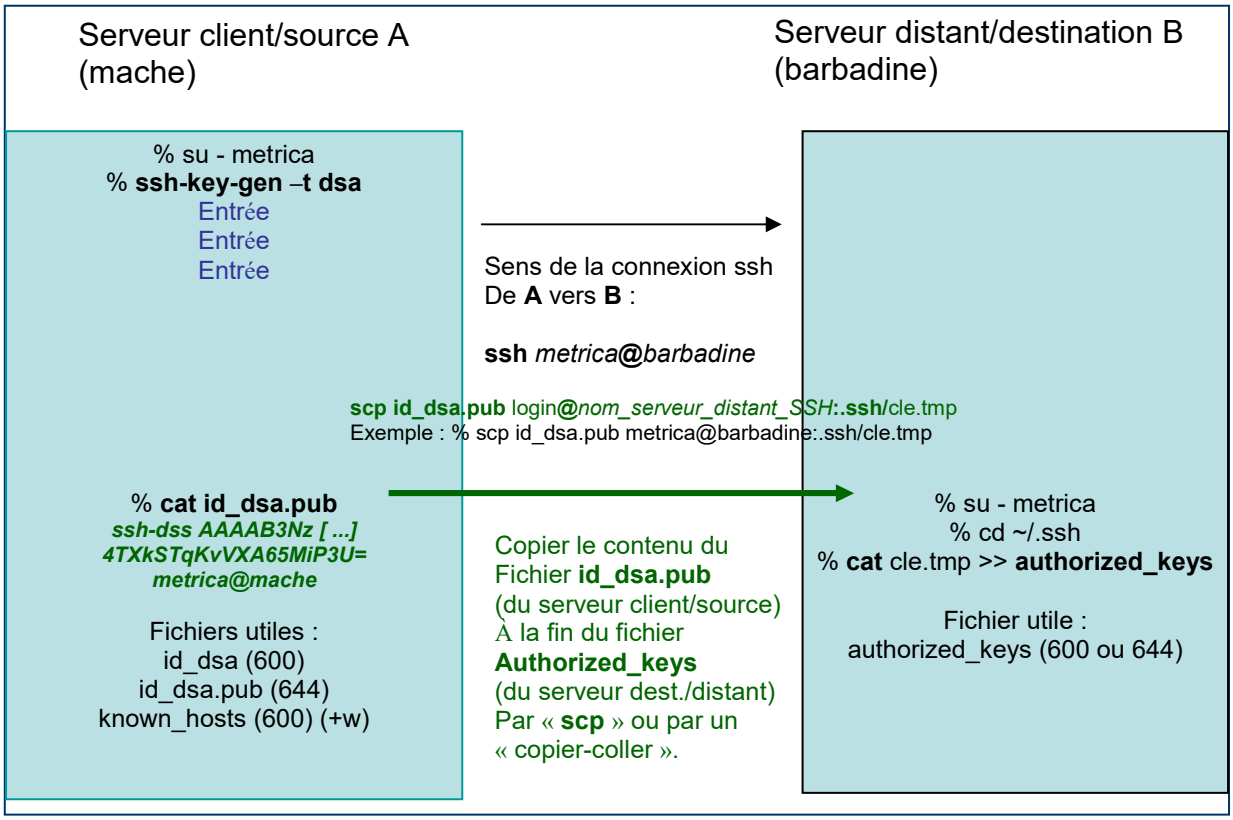
% ssh login**@** *nom_serveur_distant_SSH* (exemple : % ssh *metrica@barbadine*).

Il faut rentrer votre mot de passe de votre login « login » sur le serveur distant SSH « *nom_serveur_distant_SSH »* (c'est normalement la dernière fois que vous le saisissez).

3) mise en place de la clé publique sur sur le serveur distant SSH *nom_serveur_distant_SSH* :

% **cd ~/.ssh**
% **cat** *cle.tmp* **>> authorized_keys**

[9] Avec Debian, la commande sera :
`% ssh-copy-id -i ~/.ssh/id_dsa.pub login@nom_serveur_distant_SSH`



Le but du jeu étant d'ajouter la clé publique de votre login sur le serveur client SSH dans la liste des clés reconnues, par le serveur distant SSH « *nom_serveur_distant_SSH* ».

Exécuter la commande, depuis le serveur client SSH :

% **ssh** *login@ nom_serveur_distant_SSH*

Vous devez voir apparaître un dialogue, comme ci-après :

```
The authenticity of host 'su0245 (10.43.213.10)' can't be established.
DSA key fingerprint is fd:bc:f2:a1:a7:ff:b1:ef:25:da:69:48:0e:46:c4:c1.
Are you sure you want to continue connecting (yes/no)?

                -> Vous saisissez 'yes' : yes
                -> Le dialogue continue ainsi :

Warning: Permanently added 'su0245,10.43.213.10' (DSA) to the list of known hosts.
#######++
###---
edsprai@su0245's password:

                -> Vous saisissez le mot de passe : XXXXX

                -> Le dialogue continue ainsi :

Last login: Wed Oct 24 16:24:06 2007 from 10.40.228.8
Sun Microsystems Inc.   SunOS 5.10      Generic January 2005
No news.
%
```

Dès que vous êtes logué sur le serveur distant, vous faites Ctrl-D ou exit, pour revenir sur le serveur client. Puis vous relancez 2 autres fois :

% **ssh** *login@ nom_serveur_distant_SSH*

=> jusqu'à ce que plus aucune question soit posée et que la connexion soit transparente, comme ci-après :

```
% ssh login@ nom_serveur_distant_SSH
============================================================
  - SERVEUR D'ADMINISTRATION BT-SERVEUR - PRA - SAUVEGARDES SOCLE - 2
============================================================
Last unsuccessful login: Tue Oct 16 19:19:51 2007 on ssh from su0349svg2.sfr.com
Last login: Fri Oct 26 16:48:07 2007 on /dev/pts/1 from 10.42.223.254
%
```

A partir de maintenant, l'exécution de la commande « ssh » ne doit plus vous demander aucun mot de passe.

8 Droits sur les fichiers utiles à la connexion ssh

Note : après avoir ajouté le contenu du fichier de la clé publique à « **~/.ssh/authorized_keys** » sur le site distant, le mode de ce fichier « **authorized_keys** » devrait être mise ensuite à **600**.

Fichier ou répertoire	**Droits / mode**	**Localisation (serveur et répertoire)**
authorized_keys	600 (ou 644)	Serveur distant (ou destination/miroir) (.ssh)
.ssh	700	Serveur distant et client (ou source)
id_dsa	600	Serveur client (ou source) (.ssh)
id_dsa.pub	644	Serveur client (.ssh)
known_hosts	600 (+w)	Serveur client (.ssh)
ssh_host_rsa_key	*600*	*Serveur client (etc)*
ssh_host_rsa_key.pub	*644*	*Serveur client (etc)*
sshd_config	*644*	*Serveur distant (etc)*
dsh_config	*644*	*Serveur client (etc)*

Droits sur les fichiers utilisés par la connexion « ssh ».

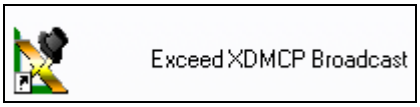
Note : ATTENTION ce mécanisme n'est pas symétrique ! Si vous avez mis en place des clés pour vous connecter sans mot de passe du serveur A au serveur B, vous ne pouvez pas vous connecter sans mot de passe de B à A (il faut alors refaire des clés dans l'autre sens).

9 Utilisation de ssh avec x11 (x-windows) et PuTTY

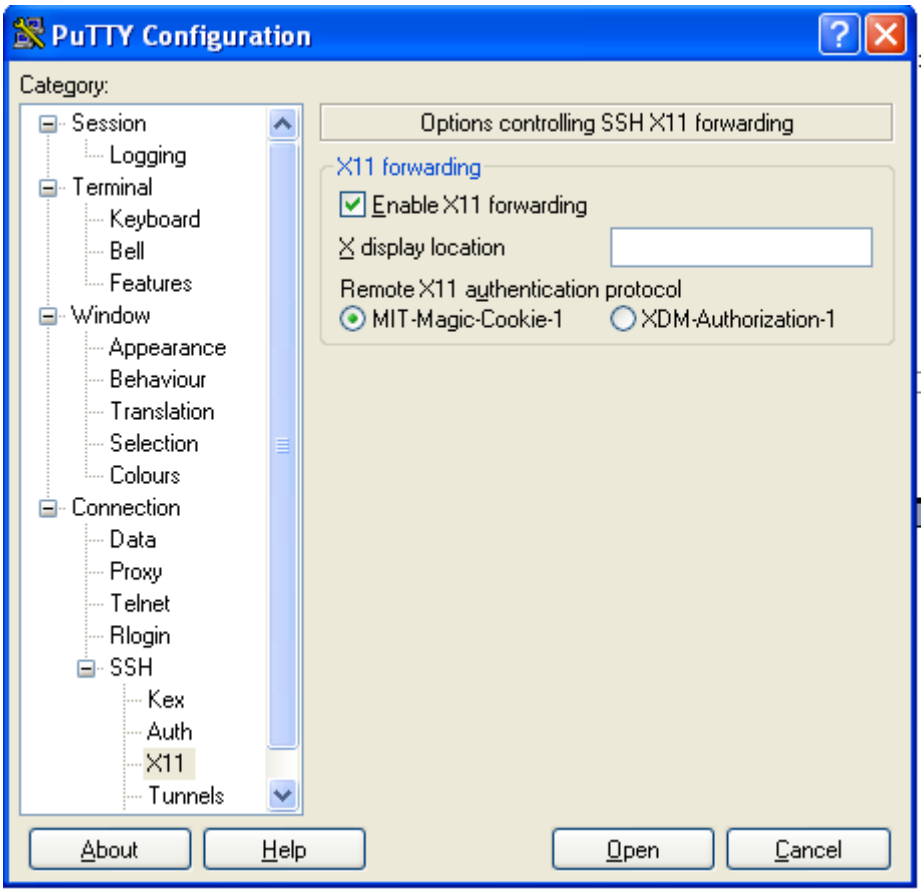
1) Dans le fichier « /usr/local/etc/sshd_config », il faut que la ligne contenant « X11Forwarding » soit 1) décommentée et 2) mise à la valeur « yes », comme dans l'exemple ci-après :

X11Forwarding yes
#X11DisplayOffset 10
#X11UseLocalhost yes

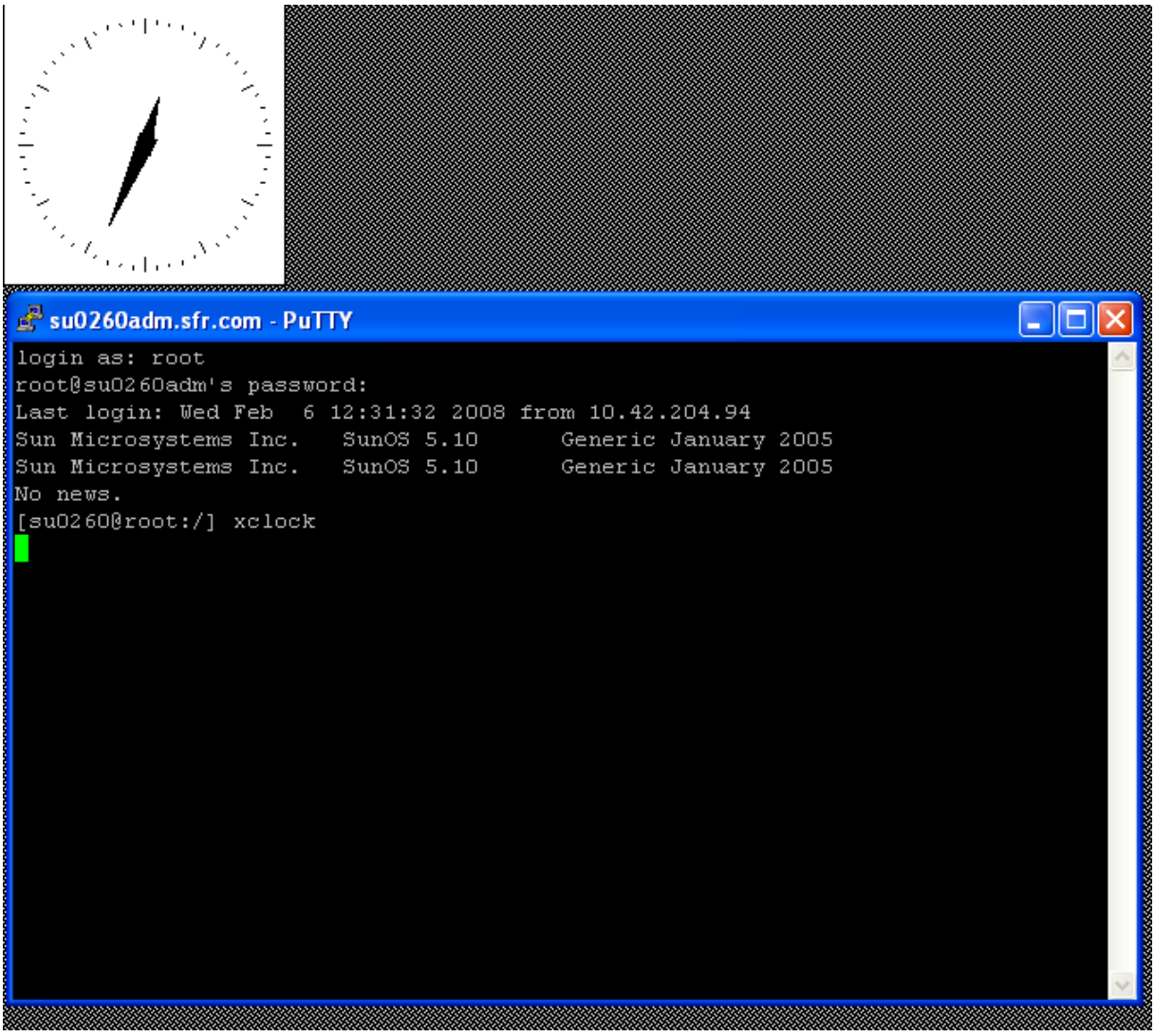
2) Il faut que vous lancez votre connexion X11, par exemple avec l'émulateur Exceed :



3) Dans l'option X11 de l'émulateur PuTTY, faut cocher la case « Enable XX1 forwarding », comme ci-après :



4) Ensuite lancer votre connexion ssh, par PuTTY. Et vous devez avoir un écran comme celui-ci :



10 Annexe : Exemple procédure de recopie de la clé publique par copier-coller

Exemple d'une autre procédure de recopie de la clé publique (de votre login côté serveur source) dans le fichier « **authorized_keys** » (de votre login côté serveur destination) existe, même si elle est moins sûre et standard (voir page suivante) :

1) côté serveur client (source) :

$ **cd .ssh**
$ **cat id_dsa.pub**
<Faire un « copier » (Copy) (dans le buffer), du contenu de « id_dsa.pub »>.

2) côté serveur distant (destination) :

Connectez-vous sur le serveur destination, avec votre login « login ».
$ **cd .ssh**
$ **vi authorised_keys**
< Faire un "coller" (Paste) du contenu copié précédemment, ici >
:wq! <sauvez et sortez>
Du serveur client, lancer la connexion "ssh" :
$ **ssh** *login@server_destination*
Cela doit aussi marcher (dans la majorité des cas).

11 Annexe : Connexion ssh utilisant les adresses IP des serveurs client et destination

Ici dans cet exemple, on remplace le nom "hostname" du serveur, par l'adresse IP du serveur source ou du serveur destination dans les fichiers : "**id_dsa.pub**" et "**authorized_keys**", parce que l'ADMIN Solaris n'a pas d'alias dans le fichier « /etc/hosts » pour chaque client ou serveur.
1) Sur le serveur client, exécutez les commandes suivantes :

$ **mkdir -p $HOME/.ssh**
$ **chmod 0700 $HOME/.ssh**
$ **ssh-keygen -t dsa -f $HOME/.ssh/id_dsa -P ''**

Les fichiers "$HOME/.ssh/id_dsa" (private key / clé privée) et « $HOME/.ssh/id_dsa.pub » (public key / clé publique) vont être créés.

2) Copier le fichier "$HOME/.ssh/id_dsa.pub" sur le serveur destination (par « scp » ou copier-coller).

3) Sur le serveur destination, exécutez les commandes suivantes :

$ **cat id_dsa.pub >> $HOME/.ssh/authorized_keys2**
$ **chmod 0600 $HOME/.ssh/authorized_keys2**

4) Selon votre version de votre OpenSSH, les commandes suivantes peuvent être nécessaires :

$ **cat id_dsa.pub >> $HOME/.ssh/authorized_keys**
$ **chmod 0600 $HOME/.ssh/authorized_keys**

5) Une autre solution consiste à créer un lien à partir de **« authorized_keys2** » vers « **authorized_keys** » :

$ **cd $HOME/.ssh && ln -s authorized_keys2 authorized_keys**

6) Sur le serveur client, tester le résultat de cette opération, par une connexion ssh au serveur destination :

$ **ssh -i $HOME/.ssh/id_dsa server**

(Optionnel) Ajouter les lignes ci-dessous, dans le fichier « sshd_config » (ou $HOME/.ssh/config ?), sur le serveur client (à vérifier) :

Nom_server_distant
IdentityFile ~/.ssh/id_dsa

12 Annexe : Résolution problèmes connexion ssh

(Paragraphe à compléter, en construction)

12.1 Problème commandes « ssh » non reconnues :

=> alors création de liens symboliques suivants sur le serveur client SSH :

```
% cd /bin
```

- a) cas où les binaires sont dans /usr/local/bin :

```
% ln -s /usr/local/bin/ssh ssh
% ln -s /usr/local/bin/scp scp
% ln -s /usr/local/bin/ssh-keygen ssh-keygen
```

- b) cas où les binaires sont dans /opt/CSIossh/bin :

```
% ln -s /opt/CSIossh/bin/ssh-keygen ssh-keygen
% ln -s /opt/CSIossh/bin/ssh ssh
% ln -s /opt/CSIossh/bin/scp scp
```

12.2 Problème de compatibilité des versions d'Openssh

Il existe deux niveaux de protocoles (les niveaux 1 (à partir de la version **3.1p1 ?)** et 2) et de plusieurs logiciels différents Openssh.
(à compléter).