

# Sommaire

|   |           |
|---|-----------|
| <b>SSH, la sécurisation par le chiffrement.....</b> | <b>1</b>  |
| Préambule.....                                      | 1         |
| La solution proposée.....                           | 2         |
| Connexion à un hôte.....                            | 3         |
| Création de paire de clefs.....                     | 5         |
| Connexion par mot de passe.....                     | 6         |
| Connexion par paires de clef.....                   | 6         |
| L'agent d'authentification.....                     | 7         |
| La copie sécurisée.....                             | 8         |
| Le transfert de fichier sécurisé.....               | 9         |
| Le tunnel et le Xforwarding.....                    | 10        |
| Autre implémentation du protocole.....              | 10        |
| Où se trouve quoi ?.....                            | 10        |
| Conclusion.....                                     | 11        |
| Bibliographie.....                                  | 12        |
| RFCs.....   | 12        |
| Logiciel.....                                       | 12        |
| <b>Copyright.....</b>                               | <b>13</b> |

# SSh, la sécurisation par le chiffrement

SSh, la sécurisation par le chiffrement

par Jean-philippe Gaulier

Licence du document : FDL

<http://www.gnu.org>

## Préambule

Il y a aujourd'hui deux possibilités de travail, en local (local work) ou à distance (remote work). Ce dernier est permis par l'utilisation du réseau, qu'il soit local (LAN) ou réellement éloigné, comme par Internet (WAN). De nombreux outils ont été fournis pour utiliser la capacité du réseau. Échanger, copier, utiliser des shells à distance. Les noms de ces outils sont respectivement FTP, rcp, telnet, etc... Bien que ces outils, utilisés pendant des années et même encore aujourd'hui dans certaines entreprises, soient très pratiques, ils comportent une faiblesse importante. Leurs transactions sont transmises en clair via le réseau. De ce fait, l'informatique devenant ouvert à un grand nombre, n'importe quelle personne mal intentionnée peut être en mesure d'observer ce que vous faites, allant même jusqu'à subtiliser vos données personnelles et mots de passe.

### Le problème par la pratique

Pour pouvoir appuyer cette notion de non-confidentialité, nous allons regarder ce qui se passe lors de l'établissement d'une connexion d'une session telnet. Le logiciel utilisé pour tracer les connexions est **ethereal**. Établissons la connexion sur un serveur *telnetd* fonctionnant sous GNU/Linux. Après quelques échanges de paquets, nous obtenons la bannière d'accueil :

```
0000 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 10 .....E.
0010 00 5f 7d 65 40 00 40 06 bf 21 7f 00 00 01 7f 00 .}_e@. @. !<.....
0020 00 01 00 17 86 10 2c fc 06 1a 2c 1a b3 30 80 18 .....ü ..°0..
0030 7f ff f7 2c 00 00 01 01 08 0a 00 23 78 ca 00 23 .ÿ+..... ..#xÊ.#
0040 78 ca 44 65 62 69 61 6e 20 47 4e 55 2f 4c 69 6e xÊDebian GNU/Lin
0050 75 78 20 74 65 73 74 69 6e 67 2f 75 6e 73 74 61 ux testi ng/unsta
0060 62 6c 65 20 63 65 6e 74 61 75 72 0d 0a ble cent aur..
```

on constate que la connexion s'effectue sur le port 23 (en 22 et 23 dans la trame).

Les données quant à elles commencent clairement en 42, avec la bannière d'accueil :

Debian GNU/Linux testing/unstable centaur

Les données ont été envoyées en clair et la bannière permet de définir le système d'exploitation sur lequel nous sommes accueillis. Continuons notre étude, la demande d'identification est proposée par le système distant :

```
0000 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 10 .....E.
0010 00 43 7d 66 40 00 40 06 bf 3c 7f 00 00 01 7f 00 .C}f@. @. !<.....
0020 00 01 00 17 86 10 2c fc 06 45 2c 1a b3 30 80 18 .....ü .E,°0..
0030 7f ff 66 8b 00 00 01 01 08 0a 00 23 78 ce 00 23 .ÿf..... ..#xÎ.#
0040 78 ce 63 65 6e 74 61 75 72 20 6c 6f 67 69 6e 3a xÏcentaur login:
0050 20
```

Il faut maintenant se faire reconnaître auprès du système. Nous supprimerons tout echo local doublant l'envoi de caractères, un pour le système distant et un pour voir ce que nous écrivons. Il en résulte une suite d'échange avec notre interlocuteur :

## Reseau-secu-ssh

|      |                         |                         |                   |
|------|-------------------------|-------------------------|-------------------|
| 0000 | 00 00 00 00 00 00 00 00 | 00 00 00 00 08 00 45 10 | .....E.           |
| 0010 | 00 35 ca b7 40 00 40 06 | 71 f9 7f 00 00 01 7f 00 | .5É-@.@. qü.....  |
| 0020 | 00 01 86 10 00 17 2c 1a | b3 4f 2c fc 06 c2 80 18 | .....°Ü,Û,Ä..     |
| 0030 | 7f ff e7 e2 00 00 01 01 | 08 0a 00 23 86 d6 00 23 | .ÿçâ..... ##.ö. # |
| 0040 | 86 63 6a                |                         | .cj               |
| 0000 | 00 00 00 00 00 00 00 00 | 00 00 00 00 08 00 45 10 | .....E.           |
| 0010 | 00 35 ca b9 40 00 40 06 | 71 f7 7f 00 00 01 7f 00 | .5É+@.@. q+.....  |
| 0020 | 00 01 86 10 00 17 2c 1a | b3 50 2c fc 06 c3 80 18 | .....°P,Û,Ä..     |
| 0030 | 7f ff e2 65 00 00 01 01 | 08 0a 00 23 86 de 00 23 | .ÿâe..... ##.P. # |
| 0040 | 86 d6 6f                |                         | .öo               |
| 0000 | 00 00 00 00 00 00 00 00 | 00 00 00 00 08 00 45 10 | .....E.           |
| 0010 | 00 35 ca bb 40 00 40 06 | 71 f5 7f 00 00 01 7f 00 | .5É>@.@. qö.....  |
| 0020 | 00 01 86 10 00 17 2c 1a | b3 51 2c fc 06 c4 80 18 | .....°Ü,Û,Ä..     |
| 0030 | 7f ff e1 55 00 00 01 01 | 08 0a 00 23 86 e4 00 23 | .ÿáU..... ##.ä. # |
| 0040 | 86 de 70                |                         | .Pp               |

Nous envoyons notre identifiant à la machine distante, les trames ne sont pas utilisées à l'économie puisque chaque paquet contient seulement un et un seul caractère comme données (data). On retrouve ici notre identifiant, jop, envoyé pour nous loguer. Ce qui est inquiétant, c'est que la même chose va également se produire pour notre mot de passe censé être privé et inviolable.

|      |                         |                         |                    |
|------|-------------------------|-------------------------|--------------------|
| 0000 | 00 00 00 00 00 00 00 00 | 00 00 00 00 08 00 45 10 | .....E.            |
| 0010 | 00 35 94 19 40 00 40 06 | a8 97 7f 00 00 01 7f 00 | .5..@.@. ".....    |
| 0020 | 00 01 87 20 00 17 e3 5e | dc 53 e3 2d 4b 8a 80 18 | ... ..ä^ ÜSä-K...  |
| 0030 | 7f ff 75 08 00 00 01 01 | 08 0a 00 34 5a b6 00 34 | .ÿu..... ...4Z¶.4  |
| 0040 | 50 e9 62                |                         | Péb                |
| 0000 | 00 00 00 00 00 00 00 00 | 00 00 00 00 08 00 45 10 | .....E.            |
| 0010 | 00 35 94 1a 40 00 40 06 | a8 96 7f 00 00 01 7f 00 | .5..@.@. ".....    |
| 0020 | 00 01 87 20 00 17 e3 5e | dc 54 e3 2d 4b 8a 80 18 | ... ..ä^ ÜTä-K...  |
| 0030 | 7f ff 5e 23 00 00 01 01 | 08 0a 00 34 5a c9 00 34 | .ÿ^#..... ...4ZÉ.4 |
| 0040 | 5a ba 6f                |                         | Z <sup>o</sup>     |
| 0000 | 00 00 00 00 00 00 00 00 | 00 00 00 00 08 00 45 10 | .....E.            |
| 0010 | 00 35 94 1b 40 00 40 06 | a8 95 7f 00 00 01 7f 00 | .5..@.@. ".....    |
| 0020 | 00 01 87 20 00 17 e3 5e | dc 55 e3 2d 4b 8a 80 18 | ... ..ä^ ÜÜä-K...  |
| 0030 | 7f ff 6a f1 00 00 01 01 | 08 0a 00 34 5a eb 00 34 | .ÿjñ..... ...4ZÉ.4 |
| 0040 | 5a c9 62                |                         | ZÉb                |
| 0000 | 00 00 00 00 00 00 00 00 | 00 00 00 00 08 00 45 10 | .....E.            |
| 0010 | 00 35 94 1c 40 00 40 06 | a8 94 7f 00 00 01 7f 00 | .5..@.@. ".....    |
| 0020 | 00 01 87 20 00 17 e3 5e | dc 56 e3 2d 4b 8a 80 18 | ... ..ä^ ÜVä-K...  |
| 0030 | 7f ff 5d a9 00 00 01 01 | 08 0a 00 34 5b 10 00 34 | .ÿ]@..... ...4[.4  |
| 0040 | 5a eb 6f                |                         | Zëo                |
| 0000 | 00 00 00 00 00 00 00 00 | 00 00 00 00 08 00 45 10 | .....E.            |
| 0010 | 00 36 94 1d 40 00 40 06 | a8 92 7f 00 00 01 7f 00 | .6..@.@. ".....    |
| 0020 | 00 01 87 20 00 17 e3 5e | dc 57 e3 2d 4b 8a 80 18 | ... ..ä^ ÜWä-K...  |
| 0030 | 7f ff be 45 00 00 01 01 | 08 0a 00 34 5c 4d 00 34 | .ÿ%E..... ...4\N.4 |
| 0040 | 5b 10 0d 00             |                         | [...               |

Le mot de passe est ici clairement exprimé. On peut lire bobo, la dernière trame transmet un retour chariot.

*Nota : Notre étude ne se portant pas sur le service telnet, nous ne rentrerons pas sur le détail des transmissions et considérons que ces quelques remarques suffisent pour notre étude.*

De la même manière, tout comme on vient de vous dérober votre mot de passe, on peut récupérer vos conversations, vos fichiers. La confidentialité est rompue, la porte est ouverte à l'insertion non-chalente de personnes non autorisées dans le système...

## La solution proposée

Afin de ne plus être ouvert au monde entier, nous avons la possibilité de chiffrer nos échanges. Pour ce faire, nous allons installer un client ssh, **Openssh**. Client libre et gratuit. Par la même occasion, vous pourrez installer le serveur, afin que votre ordinateur puisse aussi être contacté par d'autres utilisateurs, vous permettant de délivrer des services sécurisés.

Nous allons continuer notre étude dans le cadre de ce logiciel, libre à vous d'adapter à vos préférences par la suite, sachant que le principal est ici transcrit.

Afin de mieux comprendre le comportement et l'utilité de chaque programme, décomposons les différents paquets ainsi que l'usage fait des exécutables.

### Mémo

**sshd** : Serveur ssh

**scp** : Copie distante sécurisée

**ssh-keygen** : génération de clefs d'authentification, management et conversion

**sftp** : Transfert sécurisé de fichiers

**slogin/ssh** : Client ssh

**ssh-add** : Ajoute les identités DSA ou RSA à l'agent d'authentification

**ssh-agent** : Agent d'authentification

**ssh-keyscan** : Recueille les clefs publiques ssh

Nous traiterons chacune de ces applications afin de comprendre dans quels contexte elles peuvent être utilisées.

## Connexion à un hôte

Le client ssh opère selon un ordre défini :

1. les paramètres en ligne de commande
2. les informations spécifiques à l'utilisateur du fichier `~/.ssh/config`
3. la configuration globale du système dans le fichier `/usr/local/etc/ssh_config` ou `/etc/ssh/ssh_config` (cela dépendant de la distribution que vous utilisez)

**Le fichier ssh\_config** : Ce fichier représente la configuration du client ssh.

```
#$OpenBSD: ssh_config,v 1.15 2002/06/20 20:03:34 stevesk Exp $
```

```
# Ceci est le fichier de configuration par défaut des clients ssh du système. Voir  
# ssh_config(5) pour de plus amples informations. Ce fichier fournit les paramètres par défaut  
# pour les utilisateurs, ces valeurs peuvent être changées pour chaque utilisateur dans leur propre  
# fichier de configuration ou sur la ligne de commande.
```

```
# Les données de configuration sont analysées comme ci-dessous :
```

```
# 1. Les options de la ligne de commande
```

```
# 2. Le fichier spécifique de l'utilisateur
```

```
# 3. Le fichier par défaut
```

```
# N'importe quelle valeur de configuration est uniquement changée la première fois qu'elle est mise.
```

```
# Ainsi, les définitions spécifique d'hôte doivent être au début du fichier de configuration
```

## Reseau-secu-ssh

# et les options par défaut à la fin.

# Options par défaut

# Pour que les options soient prises en compte, il vous faut enlever le dièse précédent la ligne.

# Host permet de spécifier que la configuration qui suit concerne un ou plusieurs hôtes précis.

# Il est possible d'employer des caractères joker tels que \* ou ?.

**# Host \***

# Spécifie si la connexion à l'agent d'authentification doit être renvoyé vers la machine distante

**# ForwardAgent no**

# Autorise ou non la redirection du serveur graphique

**# ForwardX11 no**

# Autorise la méthode d'authentification par Rhosts. Peu sûr.

**# RhostsAuthentication no**

# Autorise la méthode d'authentification par Rhosts sur du RSA. Ne fonctionne que dans la version

# première du protocole et nécessite un setuid root. De préférence à ne pas utiliser.

**# RhostsRSAAuthentication no**

# Méthode d'authentification par RSA, ne fonctionne qu'avec la première version du protocole. Il

# faut que le fichier identity existe.

**# RSAAuthentication yes**

# Autorise la connexion par mot de passe, comme on pouvait la trouver dans rlogin ou telnet.

**# PasswordAuthentication yes**

# Si l'indicateur est positionné à yes, la demande de passephrase ou de mot de passe sera annulée.

# Cette option est utilisée dans le cas où seul des scripts interviennent et où il n'y a pas d'utilisateur

# présent pour insérer le mot de passe.

**# BatchMode no**

# Vérification de l'adresse IP de l'hôte qui se connecte dans le fichier known\_hosts

**# CheckHostIP yes**

# Cette option permet de gérer l'ajout et le changement des clefs hôtes dans known\_hosts.

# Si la clef a changé, la connexion vous sera refusée, vous indiquant le motif.

**# StrictHostKeyChecking ask**

# Localisation des fichiers identity, id\_rsa, id\_dsa

Connexion à un hôte

**# IdentityFile** ~/.ssh/identity

# IdentityFile ~/.ssh/id\_rsa

# IdentityFile ~/.ssh/id\_dsa

# Port de connexion à l'hôte distant.

**# Port 22**

# Version des protocoles supportés et désirés. Ici, on préférera employer la deuxième version si elle est disponible.

**# Protocol 2,1**

# Protocole de chiffrement (ssh v1) et protocoles disponibles par ordre de préférence (ssh v2).

**# Cipher 3des**

# Ciphers aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,aes256-cbc

# Caractère d'échappement.

**# EscapeChar ~**

Il existe différentes options pour se connecter à un hôte via ssh. Libre à vous de les utiliser ou non.

**-l login** : Identifiant de l'utilisateur.

**-v -vv -vvv** : Mode verbeux, permet d'obtenir les messages de debugage plus ou moins complets (plus il y a de v, plus vous obtenez d'information, le nombre maximum étant 3).

**-1 ou -2** : version de ssh employé. Il est déconseillé d'employer la version 1 du protocole. Bien qu'aucun exploit public ne circule, les faiblesses cryptographique du protocole ont été prouvée. De plus, la version 2 est reconnue par l'IETF

**-p port** : Numéro du port distant

**Exemple d'utilisation** :

ssh -vv -l utilisateur -p port -(1|2) hôte

ou

ssh utilisateur@hôte

L'hôte distant doit avoir un serveur ssh, nommé **sshd**, qui permet la connexion.

## Création de paire de clefs

Ssh s'appuie sur des algorithmes à paire de clefs, ce qui signifie que vous disposez d'une clef publique, disponible pour tout un chacun et une clef privée dont vous gardez jalousement l'entrée. Ce système va nous permettre de nous identifier auprès des hôtes que nous désirons contacter. Il nous faut au préalable créer le trousseau.

```
$ ssh-keygen -t dsa
```

```
Generating public/private dsa key pair.
```

```
Enter file in which to save the key (/home/jop/.ssh/id_dsa):
```

```
Enter passphrase (empty for no passphrase):
```

```
Enter same passphrase again:
```

```
Your identification has been saved in /home/jop/.ssh/id_dsa.
```

```
Your public key has been saved in /home/jop/.ssh/id_dsa.pub.
```

## Reseau-secu-ssh

```
The key fingerprint is:  
4a:0b:3b:eb:ed:05:47:56:cb:23:28:d3:d7:81:69:08
```

```
$ ssh-keygen -t rsa  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/jop/.ssh/id_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/jop/.ssh/id_rsa.  
Your public key has been saved in /home/jop/.ssh/id_rsa.pub.  
The key fingerprint is:  
52:65:28:9a:8b:64:cb:b7:6e:70:75:10:d9:0a:01:d9
```

Pour les deux algorithmes (dsa, rsa), le système nous demande dans quel fichier nous désirons sauvegarder la clef. Les fichiers par défaut semblent une bonne solution. Par la suite, une passphrase nous est demandée. Celle-ci est un « mot de passe amélioré », car non limité à un mot ou une petite suite de caractères. Il faut cependant prendre des précautions, car en cas de perte de la passphrase, vous ne pourriez plus vous authentifier en tant que propriétaire authentique.

Nous allons maintenant voir **trois méthodes de connexion via ssh**.

### Connexion par mot de passe

La première des méthodes, la plus connue et la plus usitée, reposant sur le modèle employé par rlogin ou rsh; l'hôte distant demande un mot de passe pour s'assurer de votre identité.

```
$ ssh -p 222 -l root 192.168.0.1  
The authenticity of host '192.168.0.1 (192.168.0.1)' can't be established.  
RSA key fingerprint is 74:4c:57:fd:c2:2c:0d:c3:3f:09:01:7d:e8:b7:21:24.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added '192.168.0.1' (RSA) to the list of known hosts.  
root@192.168.0.1's password:  
Last login: Thu Dec 26 03:37:03 2002 from centaur  
$
```

Je me connecte au port 222 sur une machine de mon réseau local, demandant de m'identifier comme root. La machine n'est pas connue dans mon fichier **known\_hosts**. Ce fichier contient les clés hôte DSA des serveurs SSH auxquels l'utilisateur s'est connecté. Cette méthode de connexion est intéressante, mais minimise les capacités que vous avez avec ssh. Pour des connexions telles que vers un serveur CVS, un tunnel pour votre courrier, il serait fastidieux de s'authentifier à chaque fois par ce moyen.

### Connexion par paires de clef

Puisque nous utilisons des algorithmes à paire de clefs (rsa, dsa), c'est à dire composée d'une clef secrète et d'une clef publique, il faut bien que cela nous serve à quelque chose. Nous allons donc automatiser la connexion. Pour ce faire, votre hôte contient un fichier **authorized\_keys** dans le répertoire **.ssh** où vous vous connectez (en général un home directory). Il suffit de copier l'identifiant ou les identifiants de vos clefs publiques pour que vous soyez reconnu du serveur.

**Attention**, il faut que *PubkeyAuthentication* soit positionné à *yes* dans vos fichiers de configuration. Pour insérer ma clef, j'ai plusieurs méthodes à ma disposition. Je peux employer des moyens conventionnels tels que le ftp ou le mail (à l'administrateur par exemple), ou je peux le faire au moyen d'outil sécurisé. Puisque c'est notre sujet, profitons en pour donner un exemple de *scp* sur lequel nous reviendrons ultérieurement.

## Reseau-secu-ssh

```
$scp .ssh/id_dsa.pub jop@scipc-jpg:/home/jop/.ssh/dsa2connex
Warning: Permanently added 'scipc-jpg' (RSA) to the list of known hosts.
jop@scipc-jpg's password:
id_dsa.pub 100% |*****| 613 00:00
```

Mon fichier est maintenant copié sur l'hôte distant, il me reste à inclure la clef dans le fichier `authorized_keys`. Une simple commande suffira :

```
$ cat dsa2connex >>authorized_keys
```

Je peux maintenant me connecter, l'hôte distant me reconnaît. Je peux me connecter sans mot de passe et avec une passphrase si j'en ai désiré une.

## L'agent d'authentification

Nous savons maintenant nous connecter à un hôte distant connaissant notre identité par l'intermédiaire de notre clef publique. Nous avons vu précédemment que nous étions libre de mettre ou non une passphrase afin de chiffrer notre identification et compliquer l'usurpation qui pourrait en être faite [mode parano on]. S'il paraît fastidieux d'insérer un mot de passe à chaque connexion, cela l'est d'autant plus lorsque l'on doit rentrer une phrase entière. L'agent ssh est là pour nous simplifier la vie. Lancé au début de la session (terminal ou graphique), il retient la passphrase le temps où **ssh-agent** sera actif (le temps d'une session).

Pour une session en mode terminal :

```
$ssh-agent screen
$ssh-add
```

Après avoir lancé l'agent ssh, on ajoute la passphrase à l'agent par l'intermédiaire de **ssh-add** ; tous les hôtes distants disposant de votre clef publique ne vous demanderont alors plus la passphrase puisque gérée par l'agent ssh.

De même, en mode graphique : `$ssh-agent startx`

Il vous suffit ensuite d'ouvrir un terminal et de taper

```
$ssh-add
```

L'agent sera actif pour toutes les applications utilisées en mode graphique.

Une dernière méthode consiste à **lancer l'agent-ssh** qui fournit un certain nombre de variables à exporter, puis demander l'action de `ssh-add` : `$ssh-agent`

On récupère les variables fournies et on les exporte :

```
SSH_AUTH_SOCK=/tmp/ssh-XXy0hiXW/agent.2019; export SSH_AUTH_SOCK;
SSH_AGENT_PID=2020; export SSH_AGENT_PID;
$echo Agent pid 2020;
```

On ajoute : `$ssh-add`

Pour tuer l'agent ssh, il suffit de faire :

```
$ ssh-agent -k
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
```

```
echo Agent pid 2020 killed;
```

### Invalidité des clefs d'un hôte connu :

Comme nous l'avons vu précédemment, lors de votre connexion, il vous est possible d'enregistrer l'emprunt (fingerprint) fournit par le serveur distant. Dans le cas où celle-ci serait modifiée, vous seriez immédiatement prévenu. Cela peut découler de quatre choses :

- Vous vous êtes fait pirater et l'on a modifié les clefs présentes dans votre known\_hosts.
- Le serveur distant c'est fait pirater et la valeur de ses clefs a changé.
- La troisième peut correspondre à une attaque en bon et due forme d'un man-in-the-middle.
- La dernière et néanmoins plus plausible des solution nécessite que l'administrateur du serveur distant ait changé les clefs.

La seule solution fiable demande de contacter l'administrateur en question et lui demander ce qu'il en est. Dans la pratique, peu de personnes usent de cette assurance de sécurité.

```
$ ssh 127.0.0.1
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
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
8b:d5:26:a3:79:ed:25:0f:3b:6f:fe:30:1f:ed:89:ae.
Please contact your system administrator.
Add correct host key in /home/jop/.ssh/known_hosts to get rid of this message.
Offending key in /home/jop/.ssh/known_hosts:13
RSA host key for 127.0.0.1 has changed and you have requested strict checking.
Host key verification failed.
$
```

## La copie sécurisée

Comme nous avons pu le voir plus haut, ssh fournit un outil de copie sécurisée en standard, sous le nom de scp pour **SecureCoPy**. Il remplace son ancêtre rcp. Son usage est très simple :

```
scp hôte_d_ou_je_veux_copier:source_copie hôte_destination:cible
```

Lorsque l'hôte correspond à la machine où vous vous trouvez, il n'est pas nécessaire de l'inscrire.

```
$scp iptables scipc-jpg:/home/jop/download
iptables 100% |*****| 27654 00:00
```

Vous pouvez également faire des copies récursives, comme nous le ferions avec n'importe quel autre utilitaire de copie.

```
$scp -r download scipc-jpg:/home/jpg/dl
img2.png 100% |*****| 277 00:00
internals.pl 100% |*****| 188 00:00
labels.pl 100% |*****| 271 00:00
images.pl 100% |*****| 624 00:00
portscanning.css 100% |*****| 891 00:00
index.html 100% |*****| 56753 00:00
```

## Le transfert de fichier sécurisé

Tout comme on peut copier des fichiers à distance par l'intermédiaire de scp, il est également possible de transférer des fichiers par l'intermédiaire d'un ftp sécurisé nommé **SecureFTP**.

Bien que l'outil soit encore trivial (pas de reprise de chargement en cas de coupure, pas d'indication du temps restant, pas de téléchargement ou de chargement récursif, ...), il vous permettra de faire toutes les manipulations basiques disponibles sur un ftp non sécurisé.

Commandes disponibles sur **sftp** :

|                              |  |
|------------------------------|--|
| cd path                      | Change le répertoire distant vers 'path'           |
| lcd path                     | Change le répertoire local vers 'path'             |
| chgrp grp path               | Change le groupe de fichier 'path' par 'grp'       |
| chmod mode path              | Change les permissions du fichier 'path' à 'mode'  |
| chown own path               | Change le propriétaire du fichier 'path' par 'own' |
| help                         | Affiche ce message d'aide                          |
| get remote-path [local-path] | Télécharge le fichier                              |
| lls [ls-options [path]]      | Affiche le listing du répertoire local             |
| ln oldpath newpath           | Crée un lien symbolique du fichier distant         |
| mkdir path                   | Crée un répertoire local                           |
| lpwd                         | Affiche le répertoire courant                      |
| ls [path]                    | Affiche le listing du répertoire distant           |
| lumask umask                 | Positionne l'umask local à 'umask'                 |
| mkdir path                   | Crée le répertoire distant                         |
| put local-path [remote-path] | Charge le fichier                                  |
| pwd                          | Affiche le répertoire courant distant              |
| exit                         | Quitte sftp  |
| quit                         | Quitte sftp  |
| rename oldpath newpath       | Renomme le fichier distant                         |
| rmdir path                   | Supprime le répertoire distant                     |
| rm path                      | Supprime le fichier distant                        |
| symlink oldpath newpath      | Crée un lien symbolique du fichier distant         |
| version                      | Affiche la version de sftp                         |
| !command                     | Exécute la 'commande' dans un shell local          |
| !                            | Sort vers un shell local                           |
| ?                            | Affiche ce message d'aide                          |

```
$ sftp jop@scipc-jpg
Connecting to scipc-jpg...
$sftp> lpwd
Local working directory: /home/jop
$sftp> lcd download
$sftp> lpwd
Local working directory: /home/jop/download
$sftp> put iptables
Uploading iptables to /home/jop/iptables
$sftp> quit
$
```

## Le tunnel et le Xforwarding

Tout au long de cet article, vous avez pu découvrir des applications connues, il nous manque cependant l'exportation des applications distantes. En effet, à l'aide de telnet, vous pouviez utiliser un logiciel non présent sur votre machine, mais présent sur l'ordinateur distant. Il suffisait de taper :

```
$export DISPLAY=mon_adresse:0.0
```

Comme par magie, l'application arrivait plus ou moins rapidement selon votre connexion sur votre écran. C'est ce qu'on appelle du **Xforwarding**. L'avantage une fois de plus d'utiliser ssh réside dans la connexion chiffrée et l'impossibilité à un agresseur éventuel de lire ce que vous faites via le réseau.

```
$ssh jop@scipc-jpg gedit
```

Dans l'exemple donné, nous demandons d'ouvrir la connexion ssh pour inscrire Gedit à l'intérieur (bloc-notes de Gnome). En fermant Gedit, nous fermerons la connexion ssh.

Pour que cela soit réalisable, n'oubliez cependant pas d'**activer l'option X11Forwarding** dans les fichiers de configuration.

Toujours dans la même idée de tunneler des applications, ssh vous permet d'encapsuler des protocoles. Cela est très intéressant lorsque vous avez recours à des protocoles tels que smtp, pop, ... Vous courriers, pour l'exemple, ne serons plus à l'indiscrétion de vos fournisseurs, ni des fouines du réseau.

```
ssh -L port_local:hôte_distant:port_distant nom_utilisateur@nom_hôte
```

Mettons que je veuille encapsuler les ports 25 (smtp) et 110 (pop)

```
$ssh -L 2025:scipc-jpg:25 jop@scipc-jpg
$ssh -L 2110:scipc-jpg:110 jop@scipc-jpg
```

Il vous suffit de configurer votre logiciel de messagerie favori aux ports 2025 pour le smtp et 2110 pour le pop pour recevoir correctement votre courrier, et ce de manière sécurisée.

## Autre implémentation du protocole

Openssh est l'implémentation la plus connue, mais n'est pas la seule. On peut également utiliser :

- Lsh, une version GNU
- Dropbear, un serveur dédié à l'embarqué
- Conch, une implémentation en python

## Où se trouve quoi ?

```
Le paquet OpenSSH fournit : /etc/ssh
/etc/ssh/moduli
/usr/bin/scp
/usr/bin/ssh-keygen
/usr/libexec/openssh
```

/usr/libexec/openssh/ssh-keysign  
 /usr/share/doc/openssh-3.5p1  
 /usr/share/doc/openssh-3.5p1/CREDITS  
 /usr/share/doc/openssh-3.5p1/ChangeLog  
 /usr/share/doc/openssh-3.5p1/INSTALL  
 /usr/share/doc/openssh-3.5p1/LICENCE  
 /usr/share/doc/openssh-3.5p1/OVERVIEW  
 /usr/share/doc/openssh-3.5p1/README  
 /usr/share/doc/openssh-3.5p1/README.privsep  
 /usr/share/doc/openssh-3.5p1/README.smartcard  
 /usr/share/doc/openssh-3.5p1/RFC.nroff  
 /usr/share/doc/openssh-3.5p1/TODO  
 /usr/share/doc/openssh-3.5p1/WARNING.RNG  
 /usr/share/man/man1/scp.1.gz  
 /usr/share/man/man1/ssh-keygen.1.gz

Le paquet **OpenSSH-client** fournit :

/etc/ssh/ssh\_config  
 /usr/bin/sftp  
 /usr/bin/slogin  
 /usr/bin/ssh  
 /usr/bin/ssh-add  
 /usr/bin/ssh-agent  
 /usr/bin/ssh-keyscan  
 /usr/share/man/man1/sftp.1.gz  
 /usr/share/man/man1/slogin.1.gz  
 /usr/share/man/man1/ssh-add.1.gz  
 /usr/share/man/man1/ssh-agent.1.gz  
 /usr/share/man/man1/ssh-keyscan.1.gz  
 /usr/share/man/man1/ssh.1.gz  
 /usr/share/man/man5/ssh\_config.5.gz

Le paquet **OpenSSH-server** fournit :

/etc/pam.d/sshd  
 /etc/rc.d/init.d/sshd  
 /etc/ssh  
 /etc/ssh/sshd\_config  
 /usr/libexec/openssh/sftp-server  
 /usr/sbin/sshd  
 /usr/share/man/man5/sshd\_config.5.gz  
 /usr/share/man/man8/sftp-server.8.gz  
 /usr/share/man/man8/sshd.8.gz  
 /var/empty/sshd

Les documentations sont répertoriées en bordeaux Les fichiers de configuration en vert Les exécutables en bleu

## Conclusion

Georges Orwell écrivait 1984 bien des années plus tôt et nommait un ennemi invisible Big Brother en avançant le slogan « *Big Brother is watching you* ». Non loin de cette réalité, nous avons atteint avec Internet

l'ouverture de l'outil informatique aux masses. Ne pouvant être en mesure de s'assurer que personne ne s'introduit dans votre intimité, le conseil reste d'être prudent avec vos données sensibles et personnelles. Oubliez donc les vieux protocoles non sécurisés et travaillez en toute sécurité...

## **Bibliographie**

### **RFCs**

**Telnet – numéro 854**

**SSH – Groupe de travail IETF**

**SSH – SCP – SFTP – Alix Mascret**

**Secure Shell – Denis Bodor**

**Formation Linux Debian – Alexis de lattré**

### **Logiciel**

**OpenSSH – Secure Shell**

**OpenSSL -- Secure Socket Layer**

**Ethereal – sniffeur**

**OpenOffice.org – suite bureautique**

**The gimp – Traitement d'image**

**Ksnapshot – Impression d'écran**

Je remercie tout particulièrement Anne pour le temps qu'elle a passé à la relecture de cet article ainsi qu'à sa correction et sa mise en forme.

Cette page est issue de la documentation 'pré-wiki' de Léa a été convertie avec HTML::WikiConverter. Elle fut créée par Jean-philippe Gaulier le 01/17/2003.

# Copyright

Copyright © 01/17/2003, Jean–philippe Gaulier



*Vous avez l'autorisation de copier, distribuer et/ou modifier ce document suivant les termes de la **GNU Free Documentation License**, Version 1.2 ou n'importe quelle version ultérieure publiée par la Free Software Foundation; sans section invariante, sans page de garde, sans entête et sans page finale. Pour plus d'informations consulter le site de l'APRIL.*