SSH

Romain Vimont

Ubuntu-Party

7 juin 2008

Plan

ubuntu-party

Plan

ubuntu-party

SSH : késako?

Secure Shell

SSH désigne à la fois :

- un protocole de communication sécurisé entre un client et un serveur ;
- le logiciel qui permet d'utiliser ce protocole.

Utilité

Son utilisation principale, la plus basique, est l'accès à un terminal distant, dans lequel on peut taper des commandes.

Installation

Serveur

Pour installer un serveur SSH :

```
sudo apt-get install openssh-server
```

Pour le démarrer :

```
sudo /etc/init.d/ssh start
```

Client

Le client SSH est présent par défaut dans *Ubuntu*.

Utilisation

Syntaxe

```
ssh [login@]serveur [-pport] [-C] [-X] [commande]
```

Exemples

```
ssh rom@192.168.0.1
ssh 82.240.197.138
ssh 82.240.197.138 "ps -ef"
ssh monpc.chezmoi.com -p1234 -C -X
ssh monpc.chezmoi.com -CXp1234
```

Nommez vos machines !

/etc/hosts

```
127.0.0.1    localhost
127.0.1.1    rom-laptop
192.168.0.1  tux
```

Exemples

```
ssh rom@tux
ssh tux
ssh tux -CXp1234
ssh tux "ps -ef"
```

Deux types d'authentification

Par mot de passe

- ☺ aucune configuration
- ☹ chaque connexion nécessite de retaper le mot de passe
- ☹ certaines fonctions peuvent nécessiter une authentification par clés (car pas de prompt)
- ☹ si quelqu'un connait votre mot de passe, il a accès à votre machine

Par clés

- ☺ demande d'une phrase de passe une seule fois par session
- ☺ double sécurité
- ☹ il faut le mettre en place

Principe des clés

Deux clés

Chaque client génère une paire de clés :

- une clé publique, que tout le monde peut connaître
- une clé privée, qui doit être protégée et ne doit jamais être envoyée sur le réseau

Ces deux clés sont mathématiquement liées.

Principe des clés (suite)

Principe

Quelqu'un prétend que c'est à lui qu'appartient une clé publique. Pour le prouver, il signe un message avec sa clé privée. Son interlocuteur peut alors vérifier avec la clé publique que le message a bien été signé avec la clé privée associée.

Sécurité

Tout la sécurité du mécanisme repose sur le fait qu'on *ne peut pas*, à partir de la clé publique, trouver la clé privée associée... en un temps raisonnable.

Mécanisme des clés pour SSH

Principe

- un serveur SSH possède une liste de clés publiques autorisées à se connecter
- le client possède sa clé privée, chiffrée par une passphrase
- lors de la demande de connexion, le client veut utiliser sa clé privée
  - si c'est la première utilisation de la clé lors de la session, l'utilisateur doit la déchiffrer, en tapant sa passphrase
  - une fois la clé déchiffrée, la passphrase ne sera plus redemandée
- l'authentification est réciproque : les clients possèdent un fichier contenant les serveurs qu'ils connaissent

En pratique

Génération de la paire de clés

Pour générer sa paire de clés publique/privée :

```
ssh-keygen
```

Deux fichiers sont générés :

- `~/.ssh/id_rsa` : la clé privée
- `~/.ssh/id_rsa.pub` : la clé publique

En pratique (suite)

Autoriser sa clé publique sur les serveurs

Il suffit de rajouter sa clé publique dans le fichier `~/.ssh/authorized_keys` du serveur.

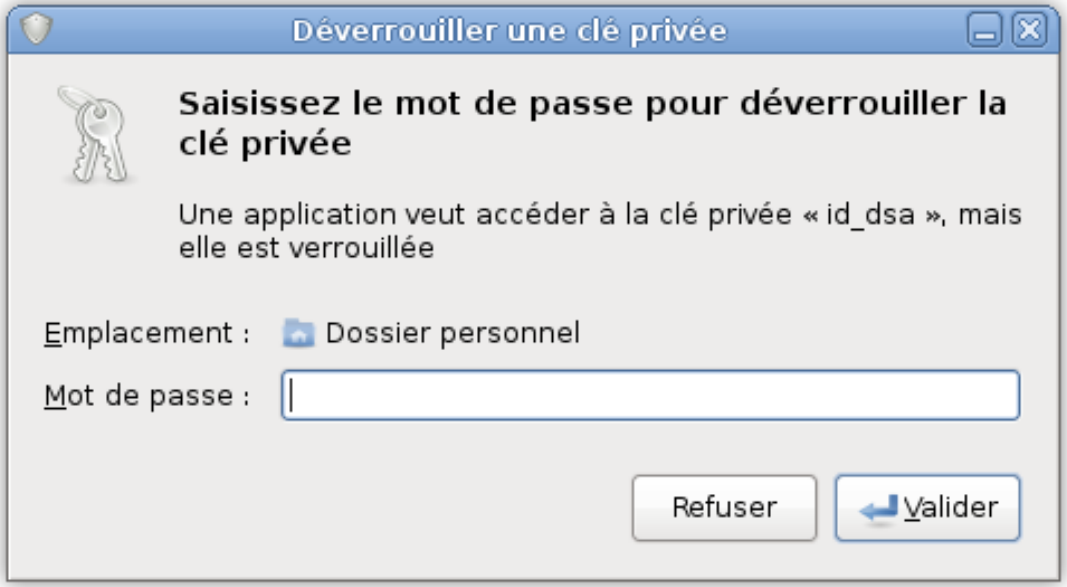
- copie sur clé usb
- recopie à la main (!)
- par SSH (avec mot de passe) !

Par SSH

```
ssh login@serveur "echo `cat ~/.ssh/id_rsa.pub` >> .ssh/authorized_keys"
```

Demande de passphrase sous Gnome

```
$ ssh login@serveur
```



En fait, par clé ET par mot de passe

En pratique

Si on désactive l'authentification par mot de passe, on ne pourra plus se connecter qu'à partir des PC autorisés, ce qui peut être génant.
En général, on laisse donc les deux types d'authentification sur le serveur (réglage par défaut) :

- si le client a une clé, elle est essayée ;
- sinon, le serveur demande le mot de passe.

Plan

ubuntu-party

Accès aux fichiers distants

Il est possible d'accéder aux fichiers d'une machine qui possède un serveur SSH, avec les mêmes droits d'accès que l'utilisateur avec lequel on est connecté.

Y'a le choix !

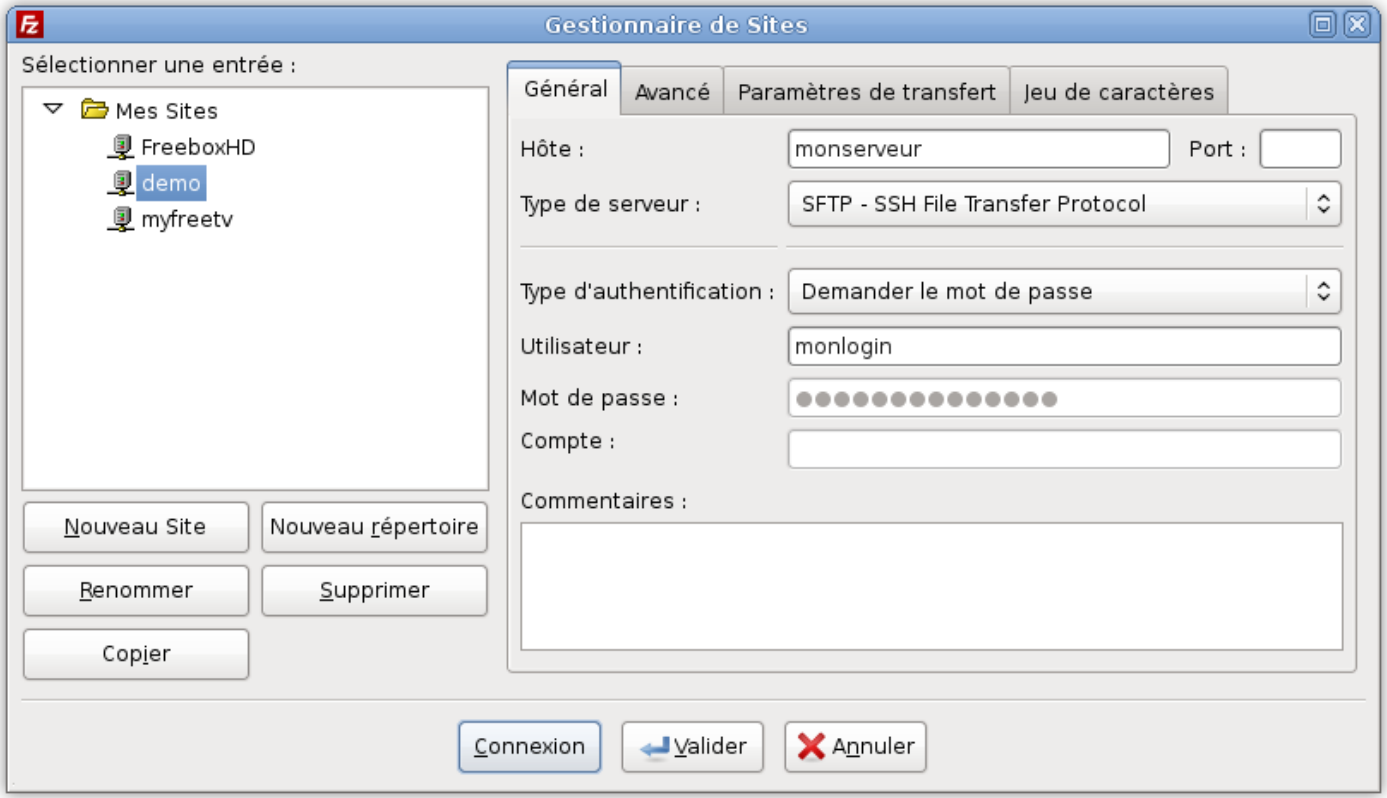
Plein de clients

- **scp** en ligne de commande, l'équivalent de `cp` pour SSH
- **sftp** en ligne de commande, l'équivalent de `ftp` pour SSH
- **Nautilus** le navigateur de *Gnome*
- **Konqueror** le navigateur de *KDE*
- **FileZilla** le célèbre client FTP, multi-plates-formes
- ...

Authentification

Certains clients ne supportent pas l'authentification par clés : dans la liste ci-dessus, **FileZilla** ne supporte que l'authentification par mot de passe.

FileZilla


Gestionnaire de Sites
Sélectionner une entrée :
Mes Sites
FreeboxHD
demo
myfreetv
Nouveau Site
Nouveau répertoire
Renommer
Supprimer
Copier
Général
Avancé
Paramètres de transfert
Jeu de caractères
Hôte :
monserveur
Port :
Type de serveur :
SFTP - SSH File Transfer Protocol
Type d'authentification :
Demander le mot de passe
Utilisateur :
monlogin
Mot de passe :
Compte :
Commentaires :
Connexion
Valider
Annuler

Accès total

Trop permissif ?

On a accès à tous les fichiers de la machine, avec les droits de l'utilisateur connecté.

Selon le cas

- si c'est pour récupérer nos propres fichiers de n'importe quel endroit, c'est pratique ;
- si c'est pour donner l'accès à quelqu'un, c'est moins bien !

MySecureShell

Objectifs

- bloquer l'utilisateur dans son *home*
- lui interdire de taper des commandes *shell*

Installation

Rajouter une source de logiciels :

```
deb http://mysecureshell.free.fr/repository/debian testing main
```

Installer le paquet :

```
sudo apt-get install mysecureshell
```

Créer un utilisateur restreint :

```
sudo adduser --home lehome --shell /bin/MySecureShell user
```

Plan

ubuntu-party

Limites du transfert de fichiers FTP-like

Diffusion d'une vidéo distante

- démarrer le transfert de fichier, pour le stocker en local
- possibilité de regarder la partie déjà téléchargée
- supprimer la vidéo une fois le visionnage terminé

Modification d'un document distant

- transférer le document en local
- le modifier localement et le sauver
- retransférer le document modifié vers le serveur

ubuntu-party

Système de fichiers virtuel

Principe

Un système de fichiers virtuel permet d'utiliser les fichiers comme s'ils étaient en local.

Diffusion d'une vidéo distante

- ouvrir le fichier vidéo
- possibilité de se déplacer "à la demande"

Modification d'un document distant

- ouvrir le document
- le modifier et le sauver

SSHFS

Principe

sshfs permet de monter un répertoire distant localement.

Installation côté client

```
sudo apt-get install sshfs
```

Utilisation

Montage :

```
sshfs login@serveur:/remote/dir /local/dir[a]
```

Démontage :

```
fusermount -u /local/dir
```

[a] `/local/dir` doit exister.

GVFS

Principe

gvfs monte la racine du serveur dans `~/.gvfs/sftp sur serveur/`.

Installation

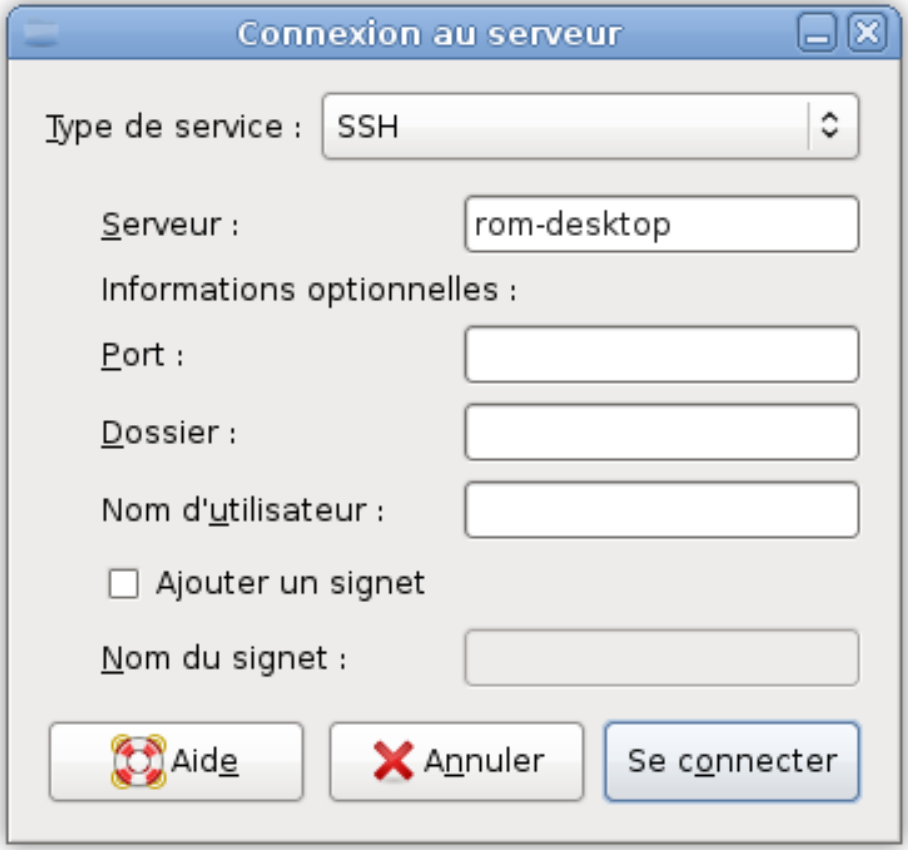
gvfs est installé et utilisé par défaut depuis *Gnome 2.22*, donc *Ubuntu 8.04*.

Utilisation

gvfs monte automatiquement un serveur lorsque l'on s'y connecte avec **nautilus** :

`sftp://login@serveur:/home/moi`

GVFS bien intégré à Gnome



SSHFS vs GVFS

SSHFS

- ☺ permet de monter n'importe quel répertoire distant dans n'importe quel répertoire local
- ☺ indépendant de l'environnement graphique (*Gnome*, *KDE*, rien...)
- ☹ montage manuel[a]
- ☹ en ligne de commande uniquement (éventuellement lanceur sur le bureau)

[a] **autofs**, très utilisé pour **nfs**, pose quelques problèmes avec **sshfs**

SSHFS vs **GVFS**

GVFS

- ☺ utilisable sans taper une seule ligne de commande !
- ☺ montage automatique lors de la connexion à un serveur
- ☺ bien intégré à *Gnome*
- ☹ que sous *Gnome*
- ☹ impossible de choisir le répertoire de montage, ni de ne monter qu'une partie de l'arborescence distante
- ☹ très récent, quelques problèmes parfois :
  - plantages de **nautilus**
  - montage automatique du répertoire ne fonctionne pas toujours

Redirection d'affichage

Syntaxe

```
ssh login@serveur -XC
```

Exécution et affichages séparés

Tous les programmes graphiques démarrés par SSH sont alors exécutés sur le serveur, mais affichés sur l'écran du client.

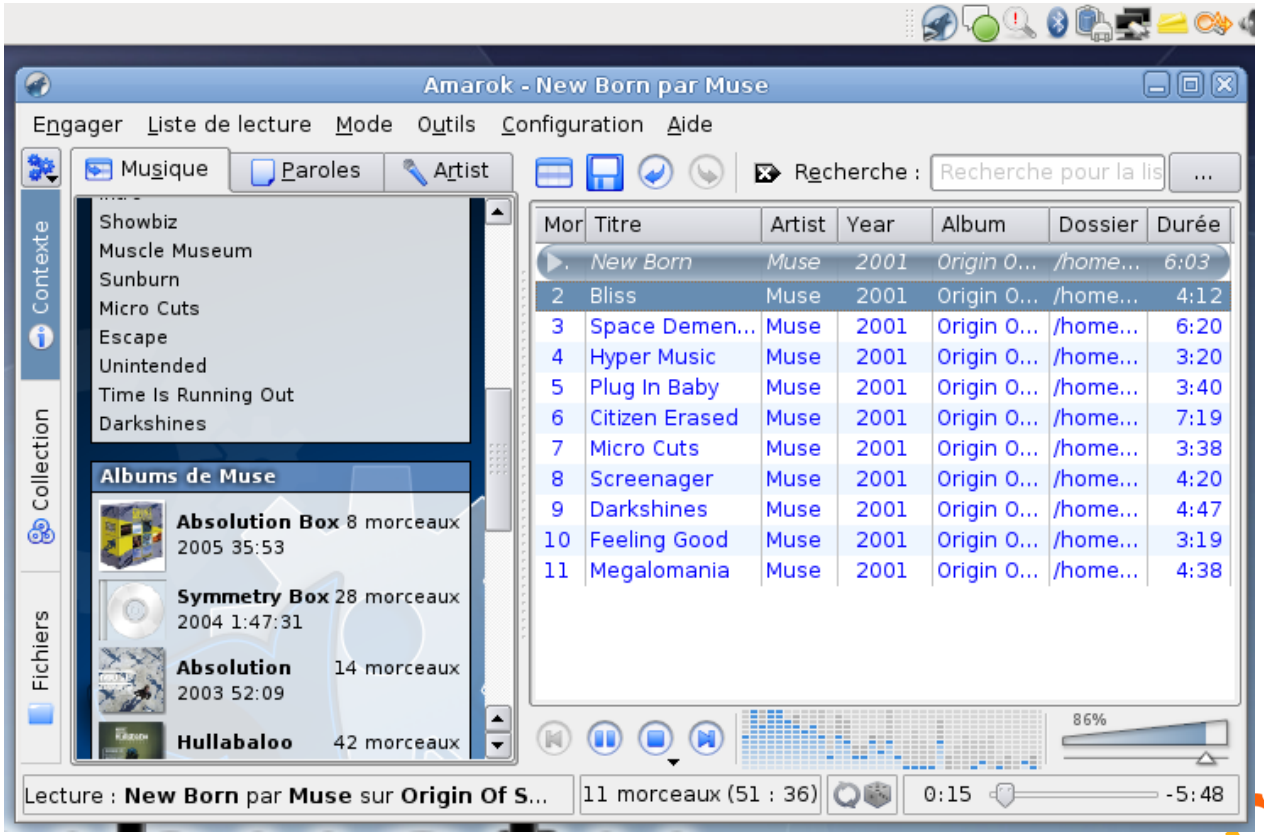
Musique !

Pour un lecteur audio lancé de cette manière :

- la musique sort sur la carte son du serveur
- le lecteur est affiché sur le client

Bien intégré au bureau

```
$ ssh login@serveur -XC amarok
```



ubuntu-party

Plus généralement

Trois entités

En fait, lorsque l'on veut écouter de la musique, il y a 3 entités :

DATA l'entité où se trouve la musique (en *Ogg Vorbis* par exemple)

SOUND l'entité où doit être lue la musique

CONTROL l'entité à partir de laquelle on veut contrôler la musique

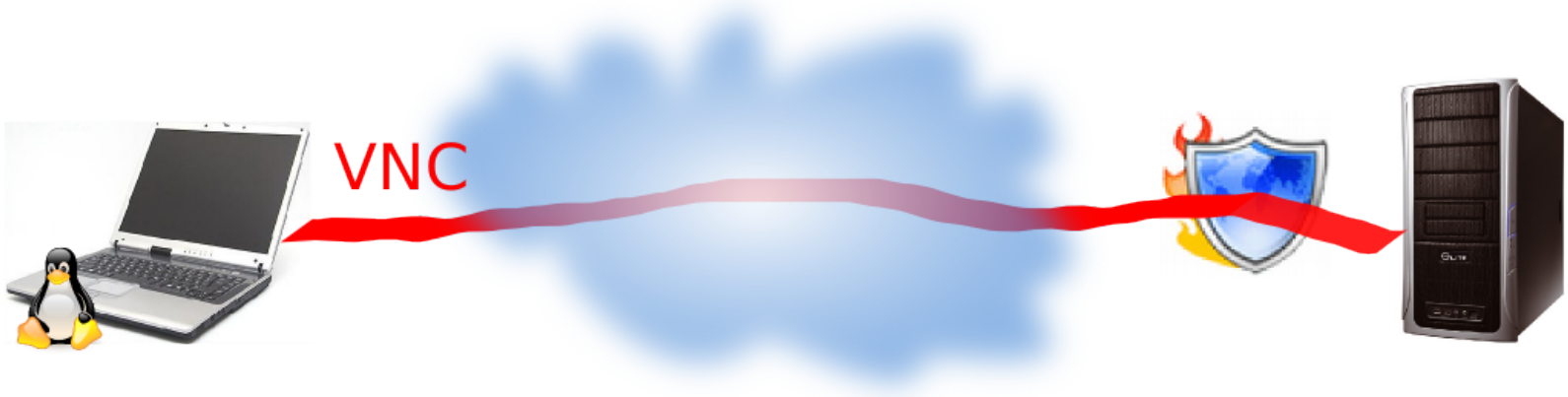
But du jeu

- amener la musique de DATA vers SOUND pour la lire ;
- amener l'affichage du lecteur de SOUND vers CONTROL.

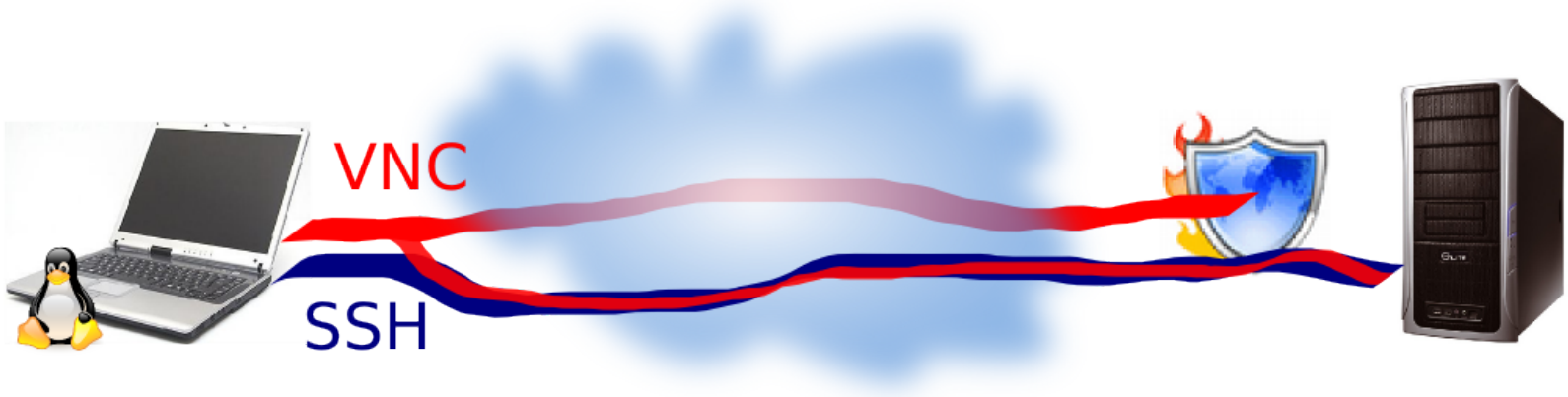
Plan

ubuntu-party

Connexion non sécurisée



Connexion sécurisée



Mise en œuvre

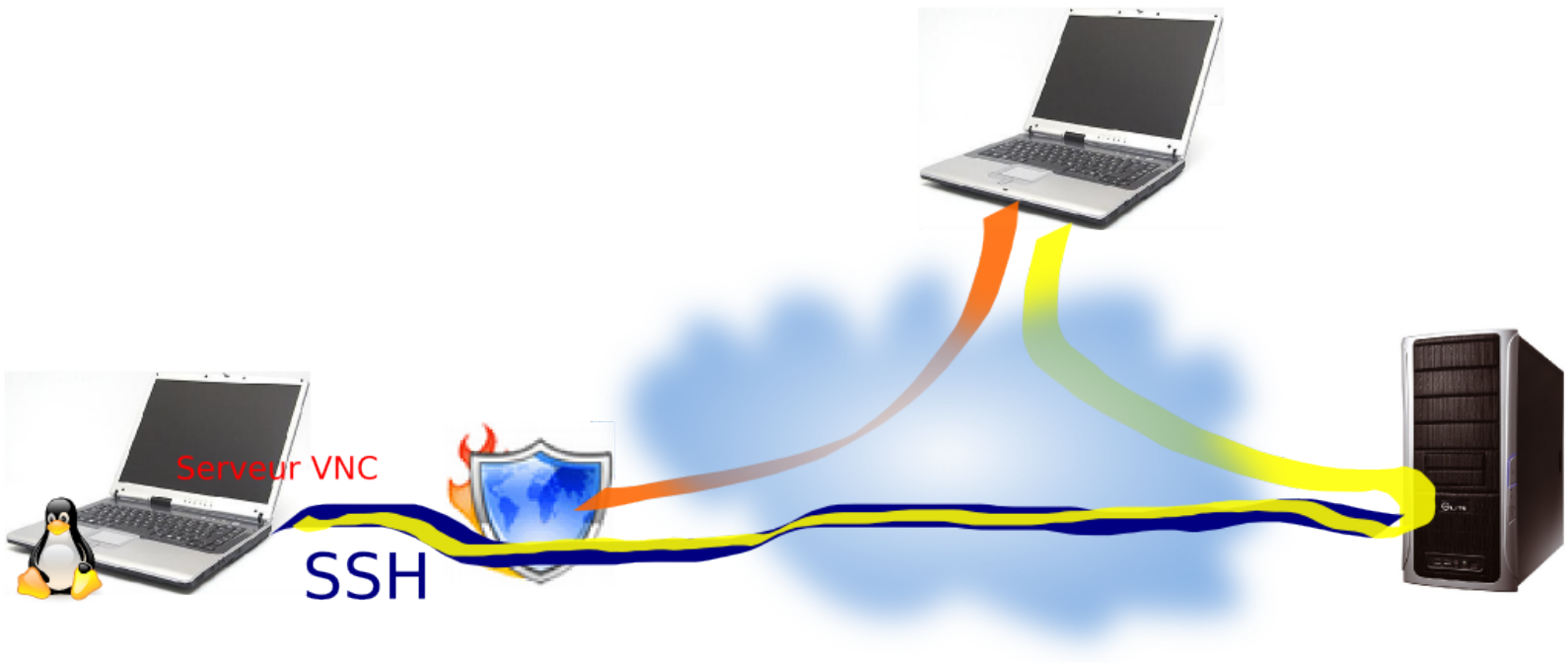
Connexion non sécurisée

```
vncviewer serveur:5900
```

Connexion sécurisée

```
ssh login@serveur -CNL1234:localhost:5900
vncviewer localhost:1234
```

Serveur derrière un routeur



Mise en œuvre

Configurer le serveur

Dans `/etc/ssh/sshd_config`, rajouter :

```
GatewayPorts yes
```

et redémarrer le serveur :

```
sudo /etc/init.d/ssh restart
```

Activer la redirection

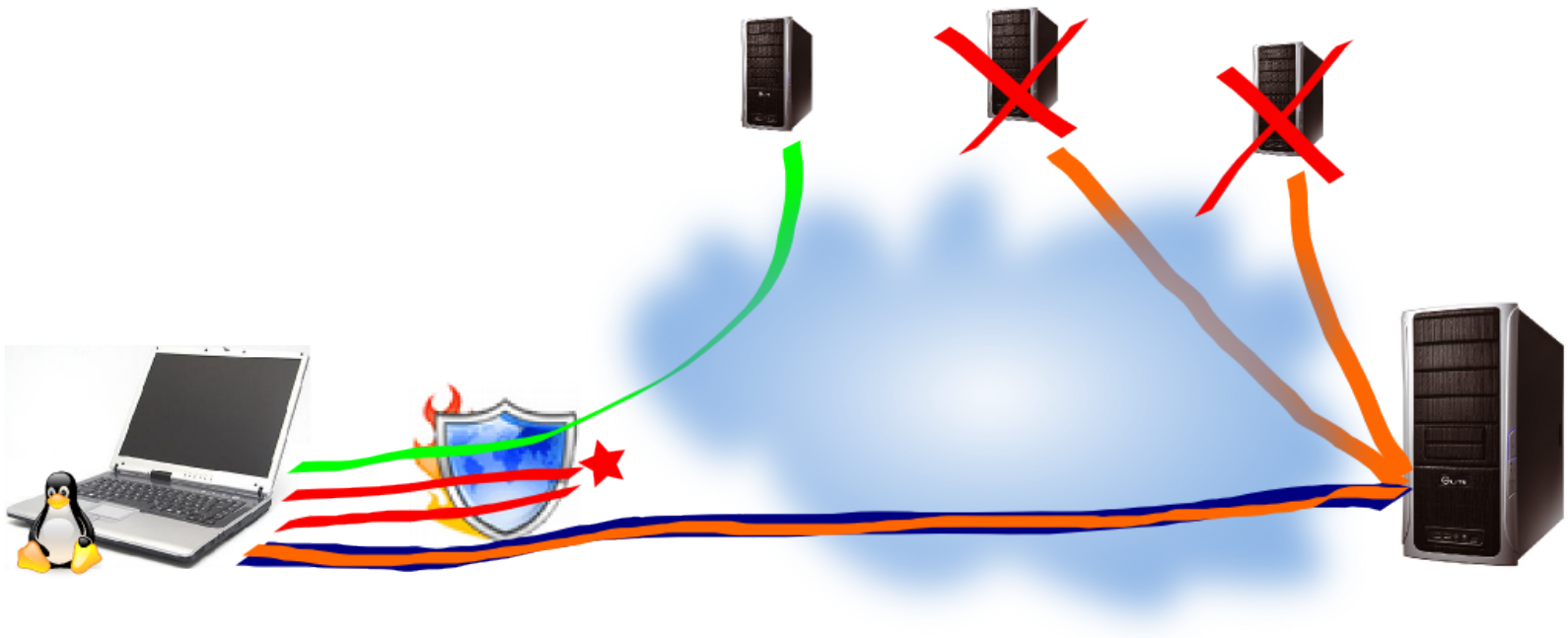
```
ssh login@serveur -CNR1234:localhost:5900
```

Connexion depuis l'extérieur

```
vncviewer serveur:1234
```

Plan

Derrière un proxy d'entreprise



Les préparatifs

Configurer le serveur

Dans `/etc/ssh/sshd_config`, rajouter :

```
Port 443
```

et redémarrer le serveur :

```
sudo /etc/init.d/ssh restart
```

Configurer le client

Installer **corkscrew** :

```
sudo apt-get install corkscrew
```

Connexion simple

Connexion SSH

```
ssh login@serveur -Cp443 -o"ProxyCommand corkscrew
proxy.entreprise.fr 3128 %h %p"
```

Simplifions-nous la vie

Associons à un *serveur* la commande *corkscrew* à utiliser. Il suffit de rajouter à la fin de `/etc/ssh/ssh_config` :

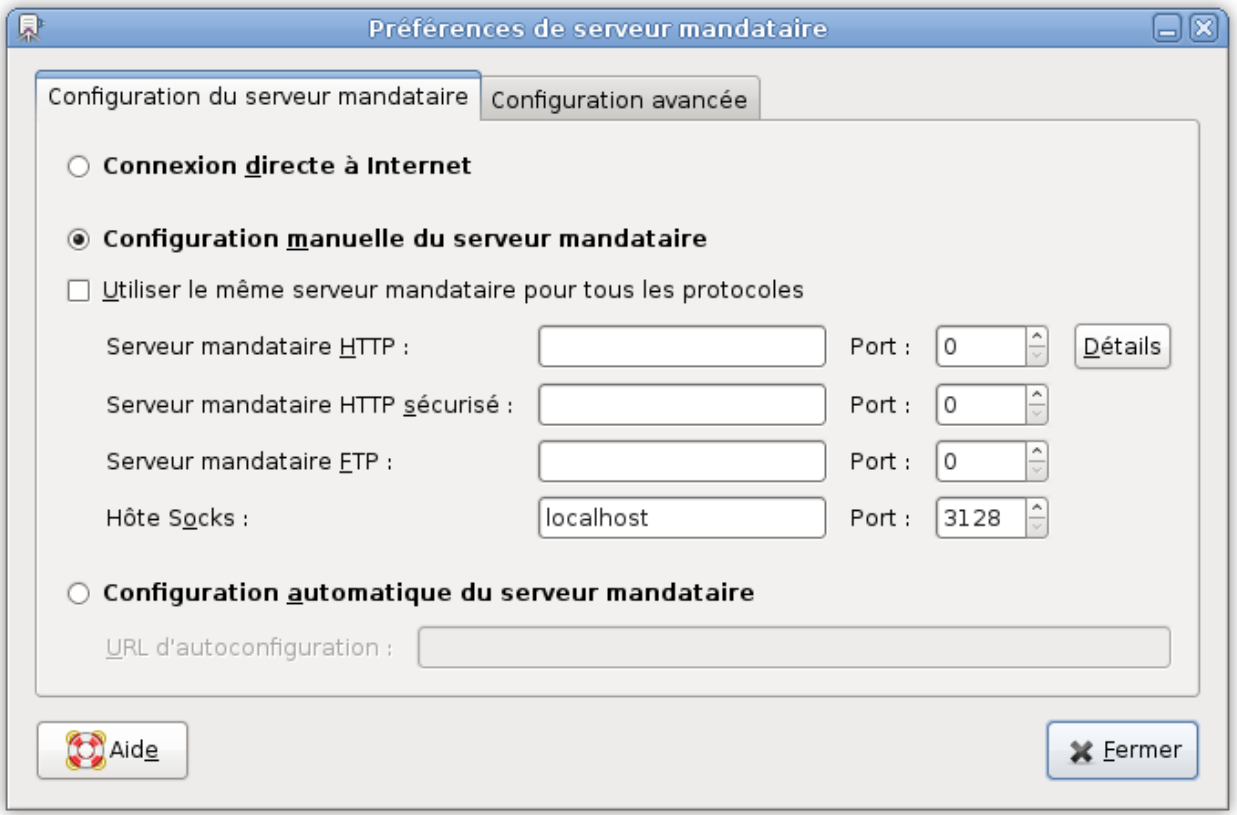
```
Host moi.mondomaine.fr
  ProxyCommand corkscrew proxy.entreprise.fr 3128 %h %p
```

Ensuite, l'utilisation est simplifiée :

```
ssh login@moi.mondomaine.fr -Cp443
```

Créer un proxy SOCK5 local

```
$ ssh login@serveur -Cp443 -ND3128
```



Ça marche

\o/