SSH Client : les Clefs du succès en environnement Unix linux et MacOS X

Philippe.Weill@latmos.ipsl.fr

26 Septembre 2012

Fortement inspiré par

Frédéric Bongat

Plan

- Base de Crypto (3 à 6)
- ssh mode simple et outils (7 à 14)
- ssh authentification forte par clé (15 à 30)
- ssh en mode tunnel (31 à 34)

le tout avec exemples et démos

Crypto (1)

On peut expliquer simplement toute la cryptographie avec les quatre fonctions suivantes

- La base de tout : le Générateur Aléatoire
  - Bouger la souris
  - Frappe au clavier
  - /dev/[u]random
- Le Condensat (empreinte, somme de contrôle,résumé)
  - Md5 , sha ....
  - Utilisé pour stocker des mots de passe unix
  - Vérification de fichier (transfert)
  - Élément de la signature électronique
  - Commande md5sum sha1sum …

```
[user@machine ~]$ sha1sum file
8976565afec640267e044381ff87163e79bcb895  file
```

Crypto (2) :Algorithmes symétriques

Principe : Une même clef est utilisée pour chiffrer et déchiffrer un message

- Appelée clef partagée,clef secrète ou clef de session
- DES , 3DES, AES, BLOWFISH, IDEA ...
- Avantage : très rapide, donc adapté pour chiffrer des flux
- Inconvénient : passage de clef entre les partenaires

Crypto(3) :Algorithme assymétrique

- Principe : Ce qui est chiffré par une clef ne peut être décodé que par l'autre
- Notion de clef privée et clef publique
- Inconvénient : c'est lent
- Rsa, dsa ...

Crypto (4) Mixte

- les applications utilisent en fait de la crypto mixte
  - échange de clefs secrètes via crypto assymétrique
  - échange de flux via crypto symétrique

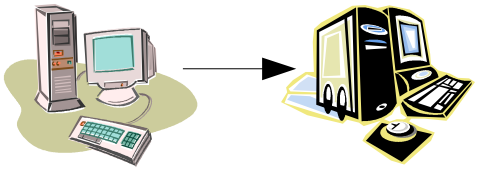
  ssh,https,imaps,pops,smtps ...
- Principe :
  - le client récupère la clef publique du serveur
  - génère localement une clef secrète (symétrique)
  - chiffre la clef secrète avec la clef publique du serveur et l'envoie au serveur
  - Le serveur la déchiffre avec sa clef privée
  - A partir de là ils ont une clef secrète en commun et peuvent chiffrer les flux via un algorithme symétrique

Introduction à SSH

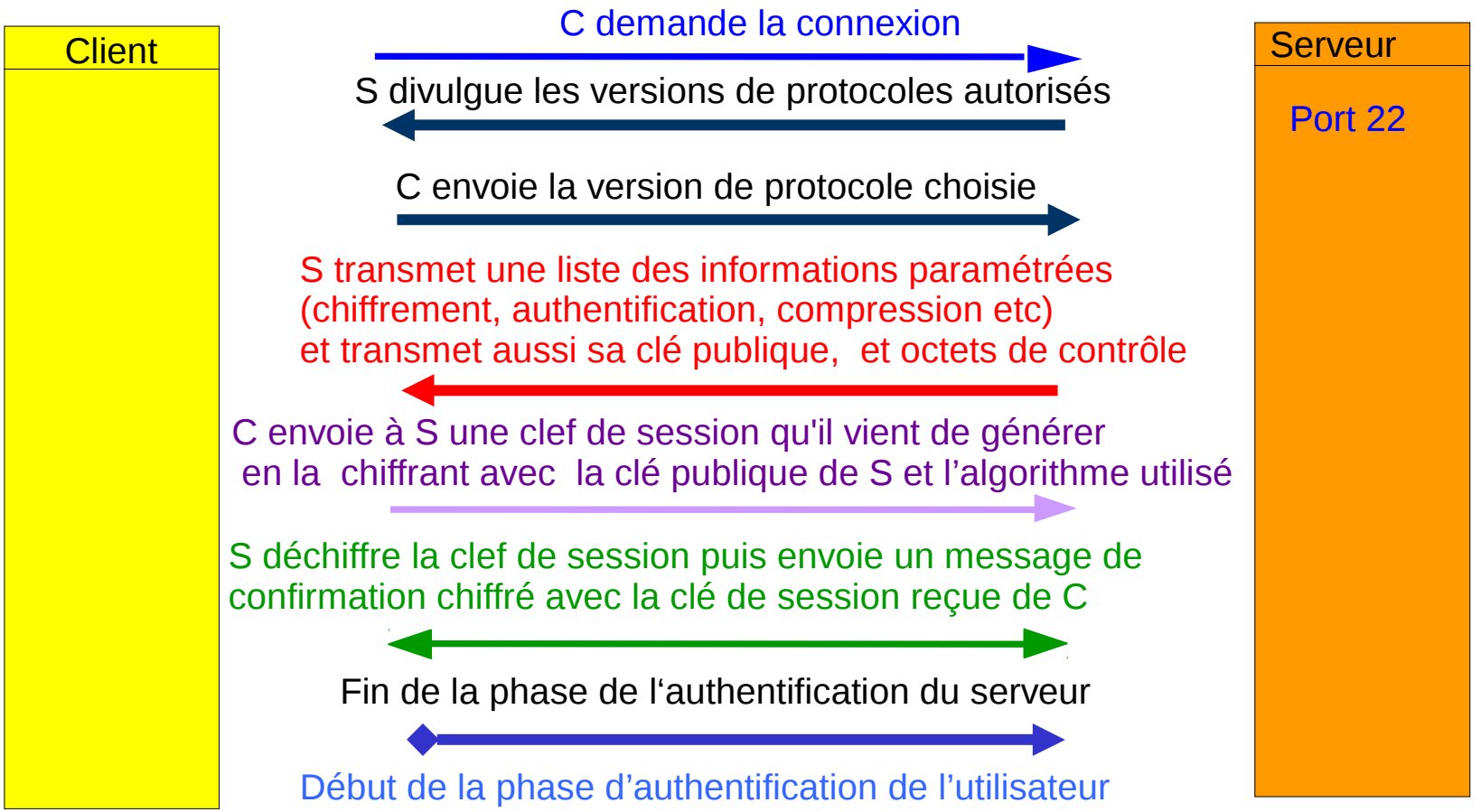
- C'est un outil de connexion à distance sécurisé à des machines en mode ligne de commande
- Il a été conçu comme remplacement sécurisé de telnet, ftp et les R-commandes
- ssh user@machine

  password :
- Fontionnement en 2 phases
  - Phase 1 : authentification des machines
  - Phase 2 : authentification de l'utilisateur
- Hautement configurable par l'administrateur et l'utilisateur
- C'est l'un des rares outils qui en ajoutant de la sécurité peut simplifier la vie de l'utilisateur

fonctionnement de ssh
Première Phase



Mise en place d'un canal sécurisé :
Tout ceci ce passe avant même la demande de mot de passe



Ssh premiers dialogues (1)

Le type de dialogue va dépendre de la valeur de l'option StrictHostKeyChecking (yes,no,ask)

la valeur par défaut de cette option dépend de l'administrateur

```
~> ssh user@machine
Warning: Permanently added 'machine,10.0.0.1' (RSA) to the list of known hosts.
Password :

~> ssh user@machine
No RSA host key is known for machine and you have requested strict checking.
Host key verification failed.

~> ssh user@machine
The authenticity of host 'machine (10.0.0.1)' can't be established.
RSA key fingerprint is 8a:1f:39:4b:a8:d0:13:9a:cf:c3:c2:13:2d:42:9f:b0.
Are you sure you want to continue connecting (yes/no)? yes
```

ssh premiers dialogues (2)

```
~> ssh user@machine
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    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
55:3b:00:77:8f:96:40:a6:9c:c5:63:5f:ed:87:bc:94.
Please contact your system administrator.
Add correct host key in /home/user/.ssh/known_hosts to get rid of this message.
Offending key in /home/user/.ssh/known_hosts:1
Password authentication is disabled to avoid man-in-the-middle attacks.
Keyboard-interactive authentication is disabled to avoid man-in-the-middle attacks.
Agent forwarding is disabled to avoid man-in-the-middle attacks.
X11 forwarding is disabled to avoid man-in-the-middle attacks.
Password :
```

Les fichiers de config de ssh chez l'utilisateur

- Répertoire $HOME/.ssh
  - known_hosts : stocke les clefs publiques des machines auxquelles on se connecte
  - config : permet de configurer le client (n'existe pas par défaut)

```
StrictHostKeyChecking ask (utile au LMD jussieu)
ServerAliveInterval 30   (garde les connexions : Numéricable)
ForwardX11 yes                    (Pratique sur Mac pour éviter le -X)
Host climserv
  hostname camelot.ipsl.polytechnique.fr
  User phweill
weill@tsunami ~> ssh climserv
```

scp et sftp

- scp : cp à travers ssh

  scp source destination

  - Exemples :
    - `scp program.f [user@]host:prog`
    - `scp [user@]host:toto.f /tmp`
- sftp: ftp à travers ssh

```
sftp user@machine
connected to machine
sftp> get fichier
```

Exécution de commande à distance

```
user@laptop ~> ssh [user@]host ls
Password:
toto.c
fichier.txt
user@laptop ~> ssh user@server /bin/true
user@laptop ~> echo $?
0
user@laptop ~> ssh user@server /bin/false
user@laptop ~> echo $?
1
```

Rsync et SVN

- Rsync
  - Synchronisation de répertoires : peut fonctionner au dessus de ssh
  - Exemples :

    ```
    rsync -avH climserv:/data/weill/results/
    /net/nfs/tmp15/weill/results/
    ```

  - Notez bien le / à la fin des chemins (important)
- SVN
  - svn peut s'utiliser par dessus ssh
  - svn checkout svn+ssh://forge/monprojet

SSH Deuxième Phase
L'authentification de l'utilisateur

Une fois que la connexion sécurisée est mise en place entre le client et le serveur, le client doit s'identifier sur le serveur afin d'obtenir un droit d'accès.

- **Par mot de passe**: Le client envoie un nom d'utilisateur et un mot de passe au serveur au travers de la communication sécurisé et le serveur vérifie si l'utilisateur concerné a accès à la machine et si le mot de passe fourni est valide
- **Par clefs publiques (Authentification Forte)** : Si l'authentification par clef est choisie par le client, le serveur va créer un *challenge* et donner un accès au client si ce dernier parvient à déchiffrer le challenge avec sa clef privée
- **Par hôte de confiance** : système équivalent aux systèmes utilisant rhost ou hosts.equiv en utilisant les clés publiques des serveurs (souvent utilisé en interne dans les clusters)
- **Et aussi par Kerberos, SmartCard, PAM ...**

Authentification forte (AF)

- Repose sur une procédure d'identification plus complexe et différente de celle des systèmes Unix classiques *(nom et mot de passe)*
- repose sur un principe d'une paire de clés publique/privée dont la clef privée est protégée par une phrase d'identification
- **ATTENTION !** la **sécurité** de ***ssh par authentification forte*** repose alors sur la protection de la clé privée :
  - il faut **impérativement** mettre une **VRAI PHRASE D'AUTHENTIFICATION** (au moins 14 caractères)
  - on ne met pas son mot de passe
- C'est en fait plus qu'un simple mot de passe, mais une véritable phrase (moins de limitation)
  - utilisation de caractères blancs (séparateur) et d'autres
  - attention aux caractères utilisés à cause des différents types de claviers afin de pouvoir entrer la passe-phrase
- L'utilisateur s'identifiera alors sans utiliser le mot de passe de la connexion classique *(mot de passe Unix)*, qui lui circule sur le réseau, mais par sa phrase d'identification sur l'hôte local

Authentification forte

- Connexion par authentification forte
  - Utilise un algorithme très puissant pour le chiffrement *(algorithme RSA/DSA)*
  - Ainsi chaque utilisateur possède son propre jeu de clés uniques (une clé privée = secrète, une clé publique = accessible par tous)
  - Un couple de clefs est utilisable sur et depuis n'importe quelle machine
  - une nouvelle connexion nécessite uniquement l’installation de la clef privée sur le client et de la clé publique sur le serveur
  - le serveur va créer un *challenge* et donner un accès au client si ce dernier parvient à déchiffrer le challenge avec sa clef privée

    ssh user@machine

    Enter passphrase for key '/home/user/.ssh/id_rsa':

AF : gestion des clés - Unix

- Gestion des clés et agents = AF
  - Commandes liées à l'AF:
    - Création des paires de clés :
      - ssh-keygen

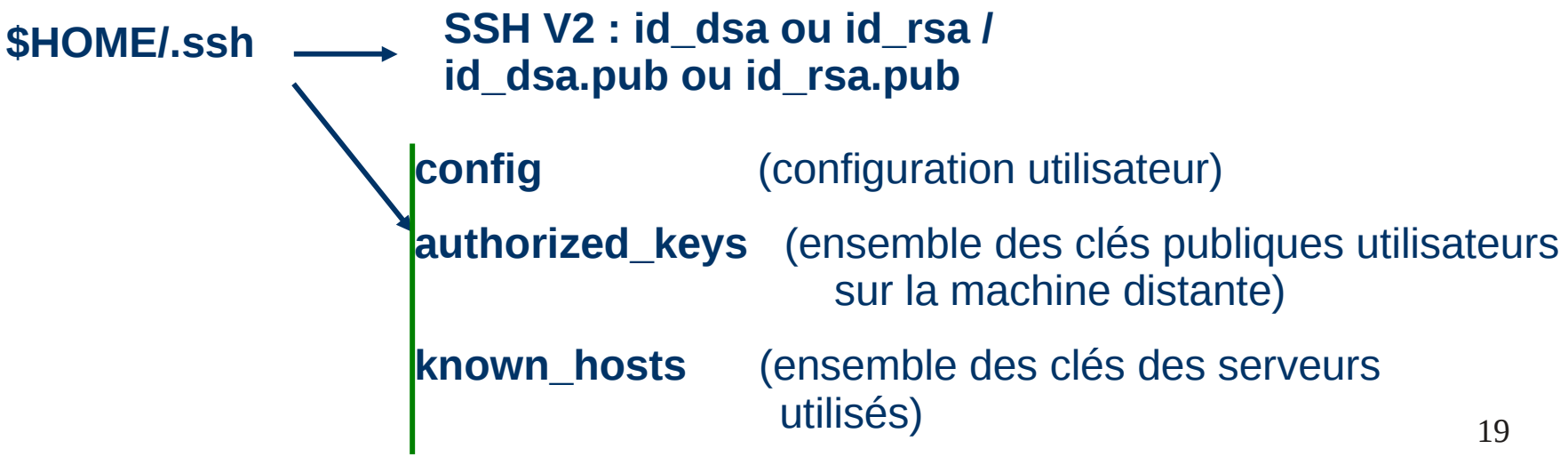
      Options : **-t** algorithme : choix de l'algorithme (**dsa** ou rsa)

      -p changer sa phrase d'identification
    - Mise en mémoire des clés (évite la saisie répétée des phrases d'identification
      - ssh-agent
    - Chargement des clés dans l'agent
      - ssh-add
  - Outils liés à l'AF
    - Keychain sous linux
    - Le trousseau pour les mac (MacOS 10.5 et > à priori)

AF : gestion des clés - Unix

Gestion des clés et agents = AF

- Structure des fichiers impliqués pour un utilisateur:



AF : gestion des clés - Unix

- Gestion des clés pour un utilisateur:
  - Création des paires de clés (SSH V2) RSA avec un exemple :

```
[root@localhost root]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase): *******
Enter same passphrase again: *******
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
26:36:03:6d:ed:28:47:c0:6c:2f:c4:d9:5a:07:fd:bc root@localhost
[root@localhost root]#
[root@localhost root]# ll .ssh
total 8
-rw-------    1 root     root          963 mar 11 23:24 id_rsa
-rw-r--r--    1 root     root          224 mar 11 23:24 id_rsa.pub
[root@localhost root]#
```

Création d'une paire de clé (v2)

phrase d'identification

Nom des fichiers de clés

ssh-keygen –t dsa

AF : gestion des clés - distribution

Distribution des clés

installation de la clef publique (id_dsa.pub) sur le serveur distant dans le fichier de stockage des clefs publiques ($HOME/.ssh/authorized_keys) :Plusieurs solutions

- transférer la clé publique par réseau (latmos)
  ceci implique un accès par mot de passe au serveur
- La mettre en place lorsque l'on est sur place (lmd jussieu)
- envoyez le fichier (clé publique) à l'administrateur du serveur distant qui l'installera sur la machine (ciclad et climserv)

Il faut donc la transférer et copier son contenu dans un fichier nommé authorized_keys sur la machine distante

Ce fichier (authorized_keys) sur la machine distante va **pouvoir contenir plusieurs clefs publiques** d'utilisateurs

Attention aux droits unix des fichiers (peut être la source de non fonctionnement de l'AF)

AF : gestion des clés - distribution

Mise en place des clés : méthode manuelle

- La clé privée côté client : *id_dsa ou id_rsa)*
- La clé publique sur le serveur dans le fichier :

  authorized_keys = clés publiques situées dans ***$HOME/.ssh***

client

serveur

$HOME/.ssh/id_dsa.pub

Transférer la clé publique sur la machine distante dans $HOME/.ssh

Puis dans $HOME/.ssh/

Faire :

cat id_dsa.pub >> authorized_keys

chmod 600 authorized_keys

AF : gestion des clés - distribution

Mise en place des clés : méthode automatique

- Utilisation de la commande : ssh-copy-id
  - installe la clé publique dans la liste des clefs autorisées (authorized_keys) d'une machine distante
  - N'existe pas sur MacOS

client

serveur

 # ssh-copy-id *serveur*

Transférer la clé publique sur la machine distante dans $HOME/.ssh et le fichier authorized_keys directement

→ **Rien à faire sur le serveur**

AF : connexion simple Unix

- Connexions par authentification forte
  - Fichier *config*

```
[root@spirou root]# cat .ssh/config
Host spip
  User fbongat
  RSAAuthentication yes
  IdentityFile ~/.ssh/id_rsa
[root@spirou root]#
```

  - Par ssh :

```
[root@spirou root]# ssh fbongat@spip
Enter passphrase for key '/root/.ssh/id_rsa':
[fbongat@spip fbongat]$
[fbongat@spip fbongat]$
```

Phrase d'authentification

  - Par sftp :

```
[root@spirou root]# sftp fbongat@spip
Connecting to spip...
Enter passphrase for key '/root/.ssh/id_rsa':
sftp>
sftp> dir
```

AF : ssh agent

- Utilisation d'un agent pour le confort
  - Va permettre d'initier des connexions sécurisées en s'authentifiant qu'une seule fois (début d'une session ou à partir d'un terminal), et ensuite de ne plus avoir à redonner sa phrase d'authentification
  - S'effectue en 2 phases :
    - Lancement d'un agent avec la commande ***ssh-agent*** dans un shell

      # ssh-agent /bin/bash
    - puis stockage en mémoire avec la commande ***ssh-add***

      # ssh-add
    - A partir de ce moment toutes les connexions lancées de la fenêtre shell ou des xterm qui en dépendent, se font sans authentification supplémentaire.

AF : ssh agent – via un shell

- Gestion des clés et d'un agent
  - **Agent dans un shell**

```
[root@spirou root]# ssh fbongat@spip
Enter passphrase for key '/root/.ssh/id_rsa':

[root@spirou root]# ssh-agent /bin/bash
[root@spirou root]#
[root@spirou root]# ssh-add
Enter passphrase for /root/.ssh/id_rsa:
Identity added: /root/.ssh/id_rsa (/root/.ssh/id_rsa)
[root@spirou root]#
[root@spirou root]# ssh fbongat@spip
[fbongat@spip fbongat]$
[fbongat@spip fbongat]$
Connection to spip closed.
[root@spirou root]#
[root@spirou root]# xterm&
[1] 2704
[root@spirou root]#
```

Phrase d'identification demandée

Lancement de l'agent

Avec le shell utilisé

Ajout de la clé en mémoire

Connexion sans identification

Toutes les nouveaux terminaux lancés (xterm) à partir du terminal dans lequel on a lancé l'agent hérite de cette propriété

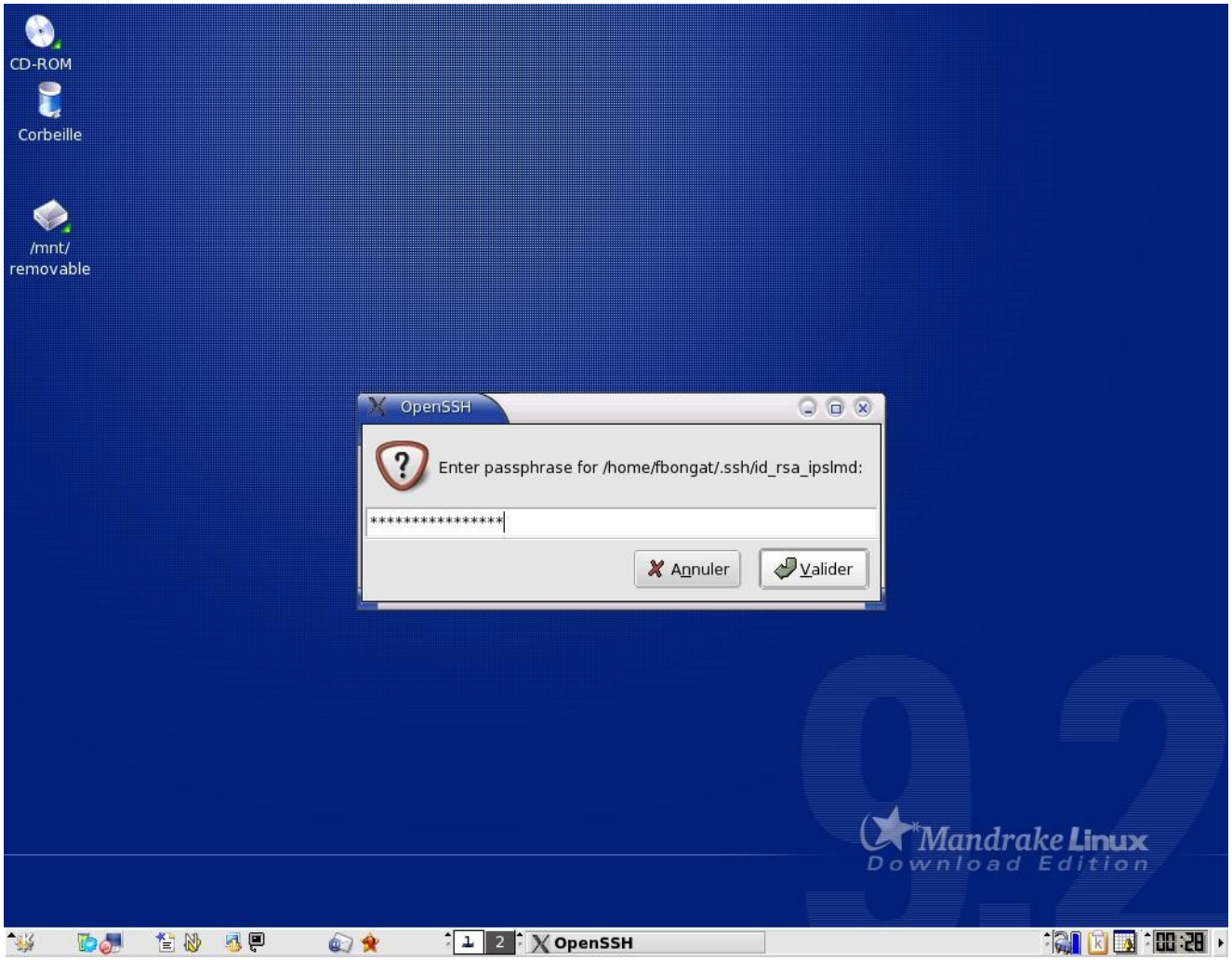
AF : ssh agent – session graphique

Gestion des clés et agents dans un environnement graphique

Très bien sur un portable
pas pratique sur une machine de bureau dans nos laboratoires(nfs)

suivant les configuration de keychain vous arriverez aux mieux à:
- demande de la passe-phrase à chaque connexion sur une nouvelle machine
- demande de la passe-phrase à la connexion à la passerelle depuis l'extérieur

Keychain gère le lancement de l'agent et peut vous demander la passe-phrase dans l'environnement graphique(kde,gnome ...)



AF : ssh agent – keychain

- **Keychain** : gestion des clés et des agents (man keychain pour en savoir +)
  - Développé par Gentoo Linux
    - Existe pour toutes les distributions (sous forme de package)
    - Installé par défaut avec tout ce qu'il faut au latmos par défaut sous linux
  - Ajout de l'AF via les scripts shell de démarrage linux
  - Le programme keychain permet de réutiliser les instances de ssh-agent + ssh-add dans des sessions différentes et, si désiré, d'inviter l'utilisateur à entrer les phrases de passe à chaque ouverture de session.
    - Package à installer sur linux *keychain*
    - Lancé automatiquement dans les scripts d'initialisation shell
      - */etc/profile.d/keychain.sh*
- Mise en oeuvre de keychain *: config recommandé par l'auteur*
  - *Sur un portable*
    - mkdir $HOME/.keychain
    - Rouvrir la session graphique , il doit normalement vous demander votre clef
  - Sur une machine de bureau(environnement NFS) :
    - mkdir $HOME/.keychain
    - Dans le fichier $HOME/.keychain/config mettre :

      KEYCHAIN_OPTIONS="--agents ssh --inherit any-once --noask"

      vous ajoutez votre clef quand vous le voulez par un ssh-add

AF : fichier de configuration

Fichier de configuration par utilisateur

- Pour forcer l'authentification forte d'un client vers un serveur lorsque celui-ci laisse tous les choix d'authentification, il suffit dans la configuration du client de configurer le fichier *config*
  - en spécifiant la variable *PubkeyAuthentication* ou *RSAAuthentication*
  - d'indiquer la clé privée utilisée (avec son path)

Fichier $HOME/.ssh/config

```
ForwardAgent yes
Host fantasio
        User fbongat
        PubkeyAuthentication yes      (ou aussi RSAAuthentication yes)
        IdentityFile ~/.ssh/id_dsa
```

AF : fichier de configuration (2)

Enchaîner deux connexions ssh en une seule commande :
ssh -t machine1 **ssh** machine2

Attention ce qui suit ne fonctionne que pour des versions très récentes d'openssh (5.4) qui n'est inclus que sur très peu de distribution linux

Donc pas utilisable aujourd'hui

On peut même faire ceci dans le fichier $HOME/.ssh/config

Exemple

Host foehn64-ext

ProxyCommand ssh -W foehn64.aero.jussieu.fr:22 sirocco.aero.jussieu.fr

Host truc.idris

ProxyCommand ssh -W user_idris@truc .idris.fr:22 user_ciclad@ciclad

Host bidule.ccrt

ProxyCommand ssh -W user_ccrt@bidule.ccrt.fr:2222 user_ciclad@ciclad

AF : mode batch ou AF sans mot de passe

Si vous avez des besoin de ce type je vous encourage fortement d'en discuter avec les administrateurs

Forçage d'une commande (ou script)

- Utilisation pour le batch : lancement de scripts de manière autonome et sans mot de passe
  - Nécessite une clé publique sans passphrase associée
    - **ATTENTION cela va à l'encontre de la sécurité !!!**
  - Il faut donc renforcer la sécurité par :
    - **Forcer l'exécution de la commande**
      - *exemple scp –rp –f fichier depuis 192.168.72.1 (machine source)*
    - **Forcer l'adresse IP émettrice pour forcer la source**
      - Sur la machine cible, dans le fichier authorized_keys

from="@192.168.72.1" **, Command="scp -rp –f fichiers_a_tranferer*" ssh-rsa AAA.....**

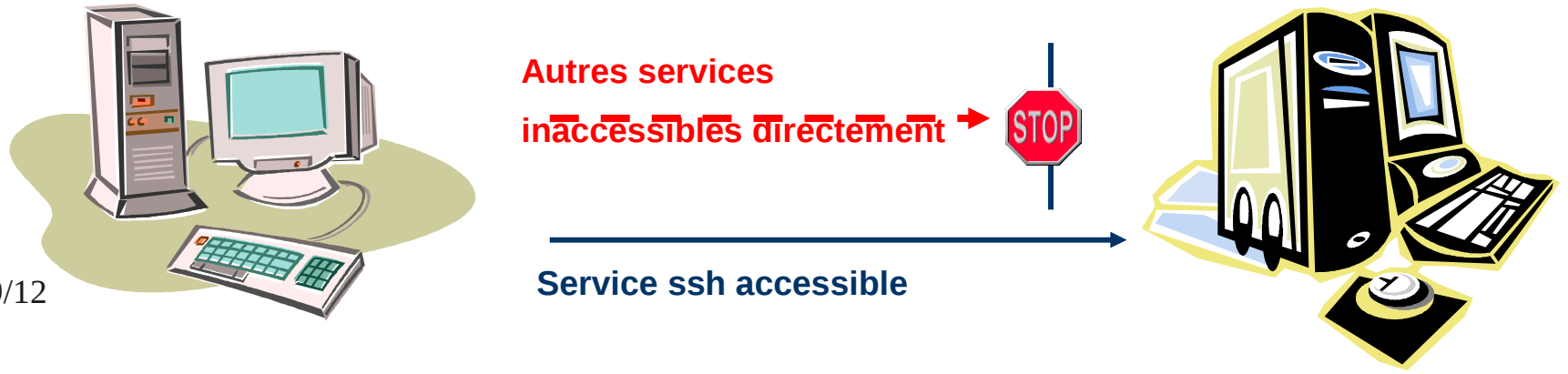
Forçage de l'IP source | **Forçage de la commande** | **Suite de la clé publique**

Tunneling

- C’est une méthode d'utilisation de SSH pour sécuriser les autres applications TCP sensibles.
  - Son principe est de créer un tunnel chiffré entre deux machines et d'y faire passer les applications dedans. Ainsi, on peut continuer à utiliser des outils comme HTTP,VNC,Bureau à distance,IMAP, POP,SMTP... de manière sécurisé
  - Ceci permet aussi l'accès à des services chiffrés mais non accessible de l'extérieur (congés , réservation de salles , changement de mot de passe ...)
- Au niveau sécurité, les tunnels peuvent poser quelques soucis car les ports locaux sur lesquels des transferts sont établis sont accessibles à tous les utilisateurs de la machine (notamment les passerelles)

Tunneling

- Transfert de port local
  - Position du problème :
    - Il s'agit donc de rediriger des services souhaités et que l'on ne peut accéder normalement car ils sont filtrés; et d'utiliser la connexion ssh (seule acceptée) pour y faire passer ces services

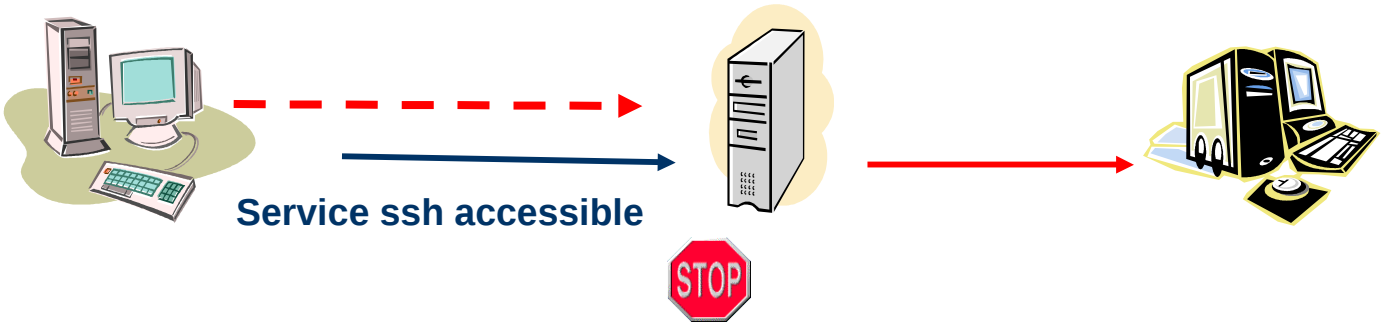


Tunneling

- Transfert de port local
  - Deux sens de la connexion possible, deux concepts :
  - Redirection locale

    ssh -L port-local:machine:port-distant machine-distante
  - Redirection distante (utilisation très rare)

    ssh -R port-distant:machine:port-local machine-distante



Tunneling : cas d'un intranet

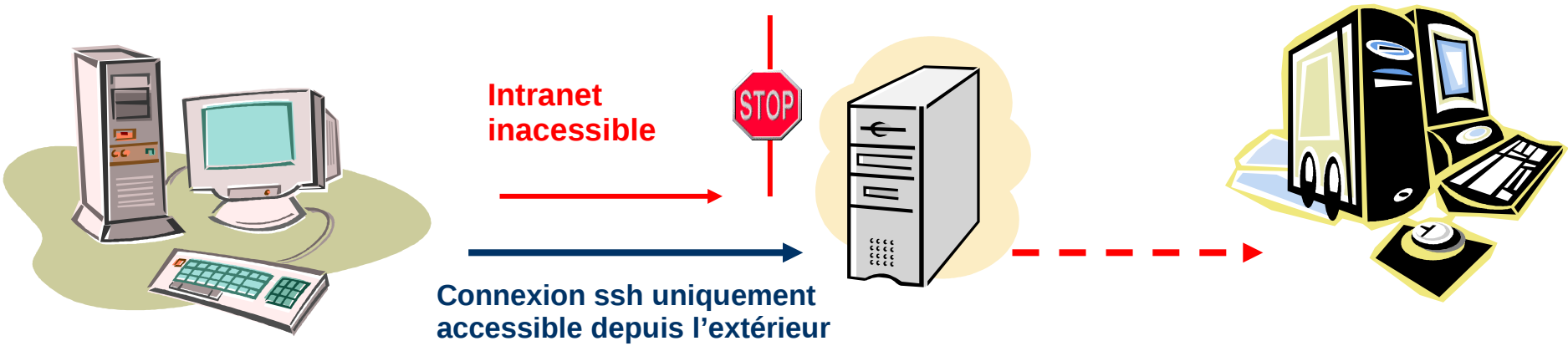
Consultation d'un Intranet depuis l'extérieur

Créer le tunnel sur la machine locale:

ssh –L 18000:ServWeb:443 passerelle

Puis lancer le navigateur Web avec l'URL : https://localhost:18000/

Principaux ports TCP : 22 ssh, 80 http, 443 https ,3389 bureau à distance,5900 vnc



Tunneling :fichier de config

Host resa

HostName sirocco.aero.jussieu.fr

LocalForward 8000 resa.latmos.ipsl.fr:443

^ attention pas de « : »

alors que l'on écrirait -L 8000:resa.latmos.ipsl.fr:443

ssh resa (ne pas fermer le shell)

dans le navigateur http://localhost:8000

Conclusion

- SSH est une boîte à outils complète
- Les connexions sont sécurisées
  - Améliore la sécurité mais peut aussi permettre de la contourner
- Existe différentes possibilités de connexions et de transferts de fichiers
- Les relais possibles des applications TCP permettent d'accéder à de nouvelles ressources
- Installé en standard sur tous les systèmes (client et serveur) Unix et MacOS X
- De très bons clients Windows notamment PuTTY, Winscp notamment en les associant à Xming
- Pour les transferts de fichiers (mode graphique) utilisez filezilla qui est disponible pour linux, Mac et windows.
  il utilise l'agent s'il est configuré