Cours de Réseau et communication Unix n°10

Edouard THIEL

Faculté des Sciences

Université d'Aix-Marseille (AMU)

Septembre 2016

Les transparents de ce cours sont téléchargeables ici :
http://pageperso.lif.univ-mrs.fr/~edouard.thiel/ens/rezo/

Lien court : http://j.mp/rezocom

Plan du cours n°10

1. Secure shell (`ssh`)
2. Client/serveur et commandes `ssh`

1 - Secure shell (`ssh`)

Outils et standards de connexions sécurisées.

Présentation de `ssh`

Ensemble de programmes et de protocoles qui permettent de se connecter sur une machine distante de manière sécurisée :

- crypte les messages, y compris les mots de passe ;
- remplace efficacement `rlogin`, `telnet`, `rsh`, `ftp` ;
- sécurise l'affichage distant X11 par un tunnel.

Historique

Origine : Tatu Ylönen, SSH Communications Security

- V 1.0 (1995) : shell, ftp, tunneling, mais problèmes sécurité
- V 2.0 (2006) : beaucoup plus sûr, meilleur ftp

Implémentations libres :

- `openssh` (1999) : openBSD, puis Linux, MacOSX, etc
- `Putty`, ... : Windows

Protocole : RFCs 4251 à 4254

Méthode de cryptage

Clé asymétrique = couple (clé publique / clé privée) tel que :

- la clé publique est connue de tous ;
- la clé privée est secrète ;
- elles sont intimement liées par une fonction mathématique ;
- il est (quasi) impossible de retrouver la clé privée à partir de la clé publique.

Méthodes principales :

- RSA : Rivest Shamir Adleman, 1977
- DSA : Digital Signature Algorithm, 1991
- ECDSA : Elliptic Curve DSA, 1992

Méthode RSA

Rivest Shamir Adleman : 1977, brevet MIT 1983, expiré en 2000.

- On choisit deux grands nombres premier et un exposant ;
- les clés sont issues d'un calcul à sens unique (inverse modulo).
- Utilise le fait que factoriser le produit de grands nombres premiers est très très long

Longueur de clé :

512 bits : cassable avec une centaine d'ordinateurs
1024 : actuellement conseillé mais cassable dans avenir proche
2048 : conseillé par les experts

Méthode DSA

Digital Signature Algorithm : 1991, breveté, utilisable gratuitement.

- On choisit deux grands nombres premier et un exposant ;
- les clés sont issues d'un calcul à sens unique (exposant modulo).
- Difficulté du problème du logarithme discret dans un groupe fini.

Longueur de clé recommandée : 2048 voire 3072 bits

Méthode ECDSA

Elliptic Curve DSA : 1992, nombreux brevets

- Cryptographie basée sur certaines courbes elliptiques.
- Clés plus courtes que RSA et niveau de sécurité supérieur ;
- chiffrement plus rapide ($20\times$), mais déchiffrer plus long ($5\times$).

Les développements théoriques sur les courbes elliptiques sont relativement récents, mais gênés par de nombreux brevets.

Utilisation des clés

Soit (P, S) une clé asymétrique, où
P est la clé publique et S la clé privée (secrète).

Pour tout message M on a : $P(S(M)) = M = S(P(M))$

- M est le message en clair ; $S(M)$ et $P(M)$ sont cryptés.
- Il faut connaître S pour décrypter $P(M)$.
- tout le monde connaît P, mais seul celui qui connaît S peut calculer $S(M)$.

Utilisation :

- Clé publique : crypter, vérifier signature, connexion ;
- Clé privée : décrypter, ou signer un document.

Dialogue crypté

Alice (P_A, S_A), Bob (P_B, S_B) et l'espionne Ennessa

Alice dialogue avec Bob : $A \xrightarrow{P_A} B$ et $B \xrightarrow{P_B} A$
Si Ennessa écoute elle ne verra que les clés publiques.

Puis Alice envoie un message secret M à Bob : $A \xrightarrow{P_B(M)} B$
Seul Bob connaît S_B et peut décoder $S_B(P_B(M)) = M$.

Réponse N : $B \xrightarrow{P_A(N)} A$; Alice décode par $S_A(P_A(N)) = N$.

Signature

Problème : Ennessa peut se faire passer pour Alice auprès de Bob, car Ennessa connaît P_B.

Solution :

Alice envoie un message secret M à Bob : $A \xrightarrow{P_B(S_A(M))} B$

Bob connaît S_B et P_A, il peut donc calculer

$$P_A(S_B(P_B(S_A(M)))) = M$$

De plus Bob a la preuve que c'est bien Alice qui a répondu car seul Alice connaît S_A.

Sous le capot, des maths

Examinons le système RSA de plus près.

- Déterminer deux entiers premiers p, q.
- Calculer $n = pq$ et $\varphi(n) = (p-1)(q-1)$.
  Facile si on connaît p et q, très compliqué sinon !
  (φ est l'*indicatrice d'Euler* de n)
- Déterminer un entier $e < \varphi(n)$ premier avec $\varphi(n)$.
- Calculer l'inverse de e modulo $\varphi(n)$, c'est-à-dire tel que

$$e.d \equiv 1 \mod \varphi(n).$$

  Facile si on connaît $\varphi(n)$, impossible sinon !

Finalement,

- (n, e) est la clef publique,
- (n, d) est la clef privée.

Sous le capot, des maths

Soit un message m. Dans le sens public $\rightarrow$ privé :

- Codage : $m \mapsto m^e \mod n$.
  On obtient le message codé c.
- Décodage : $c \mapsto c^d \mod n$.
  Comme $m^{e.d} \equiv m \mod n$ (d'après le petit théorème de Fermat), on retrouve le message d'origine.

Le sens privé $\rightarrow$ public consiste à inverser d et e.

Sous le capot, des maths

Exemple : Alice crée les clefs suivantes :

- $p = 61, q = 53$.
- $n = p.q = 3233$
- $\varphi(n) = (p-1)(q-1) = 3120$.
- On choisit $e = 17$ ce qui donne $d = 2753$ car $e.d = 46801 = 15 * 3120 + 1$.

Supposons que Bob envoie le message $m = 65$ à Alice.

- Il envoie le message $65^{2753} \mod n \equiv 588$
- Alice calcule $588^{17} \mod n \equiv 65$ et retrouve le message.

Par-dessus tout ceci, il faut ajouter les clefs de Bob.

Attaque Man in the middle

Vulnérabilité au début de l'échange : Ennessa intercepte les échanges, se fait passer pour l'autre avec sa clé (P_E, S_E) :

$$A \xrightarrow{P_A} E \xrightarrow{P_E} B \quad \text{et} \quad B \xrightarrow{P_B} E \xrightarrow{P_E} A$$

On aura par exemple le dialogue :

$$A \xrightarrow{P_E(S_A(M))} E \xrightarrow{P_B(S_E(M'))} B$$

Bob pensera que c'est Alice qui lui a écrit alors que c'est Ennessa, qui connaît M et peut le modifier (M') au vol.

Solutions :

- Échanger les clés physiquement (CD, clé USB, etc)
- Passer par une autorité de référence (certificats).

Autres attaques

- Par force brute : nécessite de puissants supercalculateurs
- Par chronométrage : en mesurant les temps de décodage de documents connus, déduire la clé
- Par chiffrage moisi : certains messages trop courts permettent de déduire la clé $\rightarrow$ salaison
- Mathématique : employer les meilleurs mathématiciens du monde
- Par porte secrète : en édictant soit-même les standards !
- etc

2 - Client/serveur et commandes `ssh`

Commandes `ssh`, `sftp`, `scp` ; serveur `sshd`

Client/serveur

Côté client : `ssh` pour ouvrir un shell ou un tunnel
`sftp` pour transférer des fichiers
`scp` pour copier des fichiers

Côté serveur : démon sshd

Installer sur Ubuntu :

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

Configuration client :	`~/.ssh/known_hosts`	machines connues
	`id_dsa.pub`	votre clé publique
	`id_dsa`	votre clé privée
	`/etc/ssh/ssh_config`	
Configuration serveur :	`/etc/ssh/sshd_config`	
	`/etc/ssh/*`	

Shell interactif

Ouvrir un shell :

```
ssh [user@]adresse
```

Pour quitter : `exit`

Par défaut, utilise le port 22 (= port d'écoute de sshd).

Changer le port :

```
ssh -p port [user@]adresse
```

Exécuter une commande à distance :

```
ssh [-p port] [user@]adresse commande [arguments]
```

Étapes du protocole

1. connexion TCP (3-way handshake) au port 22 du serveur
2. le client envoie une clef publique en utilisant la clef publique du serveur
   (s'il ne l'a pas, il doit d'abord l'accepter, ou la récupérer sur un serveur tiers.)
3. client et serveur se mettent d'accord sur une *clef symétrique*
4. les échanges suivants sont cryptés par cette nouvelle clef, y compris l'authentification de l'utilisateur
5. régulièrement (tous les 1Go transférés / toutes les heures), on change la clef.

Transfert de fichiers

Transférer des fichiers en mode interactif :

```
sftp [-oPort=port] [user@]adresse
```

Puis : `help pwd ls cd lpwd lls lcd put get quit`

Copie récursive : `put -r` ou `get -r`

Exécution d'une commande locale : `!commande arguments`

Accès au shell local : `!` puis `exit`

Copie de fichiers

Copier des fichiers à distance :

```
scp [options] [user1@host1:]file1 [user2@host2:]file2
```

Options :

`-P port`	Changer de port
`-p`	Préserver dates et mode
`-r`	Copie récursive
`-C`	Activer compression


Ne pas oublier ':' signifiant que c'est une adresse

Autorisation permanente

Par défaut, ces outils demandent un mot de passe (transmis crypté).

Problématique dans les scripts.

On peut rendre une autorisation permanente (le temps d'une session) avec `ssh-agent`.

On peut rendre l'authorisation permanente en déposant la clé publique sur le système cible avec `ssh-copy-id` (la clé privée reste sur la machine source).

Dépot de clés

Pour autoriser un client de façon permanente :

- Générer 1 clé locale (publique/privée) une fois pour toutes :
  `ssh-keygen -t dsa -b 1024`

(ou `rsa` ou `ecdsa`)

Le résultat est : `~/.ssh/id_dsa.pub` et `id_dsa` (privée)

(ou `id_rsa` ou `id_ecdsa`)

- Pour chaque machine cible, on envoie la clé publique :
  `ssh-copy-id -i ~/.ssh/id_dsa.pub user@cible`

→ La cible demandera une dernière fois le mot de passe, puis stockera la clé publique dans `~/.ssh/authorized_keys`

- On fait un essai :
  `ssh user@cible` ou `sftp user@cible` (ou `scp`)

Affichage distant

```
ssh user@cible
xclock                  Error: can't open display: :0
echo $DISPLAY           :0

ssh -X user@cible
xclock                  Ok ! S'appelle un "tunnel X11"
echo $DISPLAY           localhost:10
```

Explications : X11

• X11 est le système d'affichage graphique par défaut sur tout les Unix, qui permet affichage local ou distant.

• Toutes les applications qui veulent afficher sont des "clients X11".

• Sur l'ordinateur qui affiche, tourne un "serveur X11" qui :

- gère le display = écran/clavier/souris.
- a les bon drivers pour la carte graphique
- autorise ou rejette les clients (sécurité).

Explication : display

Un display désigné par `adresse:numéro` où `numéro` $\in 0 \ldots 63$

- Affichage local : avec socket TCP/UN sur `/tmp/.X11-unix/X0 .. X63`
- Affichage distant : socket TCP/IP sur ports 6000 à 6063

`ssh -X` ouvre un "tunnel" TCP/IP entre

- le client ssh
- le serveur sshd distant, socket d'écoute port 6010 (le display numéro 10)

Donc – `xclock` s'adresse au port 6010 de la machine distante, en fait à sshd
– sshd tunnelle au client ssh,
– le client ssh s'adresse au serveur X local, en TCP/UN,
– le serveur X local affiche.

Tunnels

On peut demander un tunnel TCP/IP crypté avec l'option -L :

```
ssh -L port_local:adresse2:port_distant user@adresse1
```

Ceci établit une connexion ssh sur `user@adresse1`, avec de plus un tunnel entre :

`localhost:port_local` et `adresse2:port_distant`

la machine `adresse1` servant de relais.

Exemple de tunnels

`me@mypc`	mon portable à l'extérieur
`bob@sas`	la passerelle d'accès au labo
`jo@lab`	mon pc dans le labo

De `me@mypc` je veux me connecter sur `jo@lab`, mais interdit directement ; faire : `me@mypc` $\longrightarrow$ `bob@sas` $\longrightarrow$ `jo@lab`

- Connection sur `bob@sas` :

```
xterm -e ssh -L 10000:lab:22 bob@sas &
```

$\rightarrow$ tunnel permanent, le terminer en fermant le `xterm`

- On peut ensuite ouvrir plusieurs connexions sur le même tunnel :

```
ssh -X -p 10000 jo@localhost
sftp -oPort=10000 jo@localhost
scp -P 10000 fichier jo@localhost:
```

Autres usages

Tunnel vers d'autres ports :

25 (SMTP)	envoi de mails
110 (POP3)	réception de mails
80 (HTTP)	serveur web privé

(configurer client `thunderbird` sur `localhost:10000`)

On peut mettre des tunnels dans des tunnels pour rebondir de passerelle en passerelle.

Exercice : que faut-il faire pour : A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ D ?

Raccourcis

me@mypc ⟶ bob@sas ⟶ jo@lab

```
$ cat $HOME/.ssh/config

Host lab22
  HostName localhost
  Port 10000
  HostKeyAlias lab22
  User jo

$ xterm -e ssh -L 10000:lab:22 bob@sas &
$ ssh lab22   # au lieu de ssh -p 10000 jo@localhost

$ man 5 ssh_config
```

Commande `rsync`

Copie de fichiers de la famille `ssh`.

Efficace pour sauvegardes : compare des empruntes MD5

```
rsync -zaibun  --backup-dir="svg/" --suffix=".back" \
      --exclude "*svg/*" --exclude "*~" \
      [user1@host1:]path1 [user2@host2:]path2
```

- `-z` compresse
- `-a` archive : récursif + préservation
- `-i` affiche résumé changements
- `-b` fait un backup
- `-u` ignore fichiers moins récents
- `-n` fait un essai sans copie