

M3102 : Services Réseaux

Bruno BEAUFILS

2020/2021

1. Introduction

Cours

Administration système

2. SSH

Secure SHell

Connection à un ordinateur distant avec accès à un terminal

- accès **distant** : shell, copie de fichier et transfert de ports
- connection **chiffrée** : cryptographie asymétrique

Histoire

- conçu pour remplacer rlogin, rsh, rcp, telnet et ftp
- débute en 1995, normalisé en 2006
- 2 versions (ssh1, ssh2)

Détails

① protocole de communication

- TCP port 22
- architecture client/serveur

RFCs **4251** (4252, 4253, 4254, 4255)

② ensembles de programmes

- implémentation du protocole

Secure SHell

Connection à un ordinateur distant avec accès à un terminal

- accès **distant** : shell, copie de fichier et transfert de ports
- connection **chiffrée** : cryptographie asymétrique

Histoire

- conçu pour remplacer rlogin, rsh, rcp, telnet et ftp
- débute en 1995, normalisé en 2006
- 2 versions (ssh1, ssh2)

Détails

① protocole de communication

RFCs **4251** (4252, 4253, 4254, 4255)

- TCP port 22
- architecture client/serveur

② ensembles de programmes

- implémentation du protocole

Logiciels

- Architecture client/serveur
 - serveur sur la machine distante
 - clients sur la machine locale
- Plusieurs implémentations
 - [OpenSSH](#)
 - issu de [OpenBSD](#)
 - licence libre
 - implémentation complète (serveur **et** client)
 - disponible sur beaucoup d'OS
 - [PuTTY](#) (client Windows)
 - Autres : [Dropbear SSH](#), etc.
- Debian

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

sshd
ssh, sftp, scp

la référence

Chiffrement à clés

- Chiffrer = rendre illisible un message
 - exemple : code de César (ROT13)
 - vocabulaire : chiffrer \neq crypter
- Chiffrer avec une **clé** = utilisation d'une clé pour chiffrer et déchiffrer
 - objectif : transmettre un message sans crainte d'une interception
 - connaître le message chiffré et le codage ne suffit pas à déchiffrer
 - la clé est
 - juste une information (un nombre par exemple)
 - un paramètre de la fonction de codage
- Chiffrement **symétrique** (à *clé secrète*)
 - la même clé sert à chiffrer et déchiffrer
 - les 2 parties doivent connaître la clé avant la communication
- Chiffrement **asymétrique**
 - 2 clés différentes :
 - une sert à chiffrer
 - une autre sert à déchiffrer
 - la clé publique peut (doit) être distribuer largement

clé publique
clé privée

Chiffrement à clés

- Chiffrer = rendre illisible un message
 - exemple : code de César (ROT13)
 - vocabulaire : chiffrer \neq crypter
- Chiffrer avec une **clé** = utilisation d'une clé pour chiffrer et déchiffrer
 - objectif : transmettre un message sans crainte d'une interception
 - connaître le message chiffré et le codage ne suffit pas à déchiffrer
 - la clé est
 - juste une information (un nombre par exemple)
 - un paramètre de la fonction de codage
- Chiffrement **symétrique** (à *clé secrète*)
 - la même clé sert à chiffrer et déchiffrer
 - les 2 parties doivent connaître la clé avant la communication
- Chiffrement **asymétrique**
 - 2 clés différentes :
 - une sert à chiffrer
 - une autre sert à déchiffrer
 - la clé publique peut (doit) être distribuer largement

clé publique
clé privée

Chiffrement à clés

- Chiffrer = rendre illisible un message
 - exemple : code de César (ROT13)
 - vocabulaire : chiffrer \neq crypter
- Chiffrer avec une **clé** = utilisation d'une clé pour chiffrer et déchiffrer
 - objectif : transmettre un message sans crainte d'une interception
 - connaître le message chiffré et le codage ne suffit pas à déchiffrer
 - la clé est
 - juste une information (un nombre par exemple)
 - un paramètre de la fonction de codage
- Chiffrement **symétrique** (à *clé secrète*)
 - la même clé sert à chiffrer et déchiffrer
 - les 2 parties doivent connaître la clé avant la communication
- Chiffrement **asymétrique**
 - 2 clés différentes :
 - une sert à chiffrer
 - une autre sert à déchiffrer
 - la clé publique peut (doit) être distribuer largement

clé publique
clé privée

Chiffrement à clés

- Chiffrer = rendre illisible un message
 - exemple : code de César (ROT13)
 - vocabulaire : chiffrer \neq crypter
- Chiffrer avec une **clé** = utilisation d'une clé pour chiffrer et déchiffrer
 - objectif : transmettre un message sans crainte d'une interception
 - connaître le message chiffré et le codage ne suffit pas à déchiffrer
 - la clé est
 - juste une information (un nombre par exemple)
 - un paramètre de la fonction de codage
- Chiffrement **symétrique** (à *clé secrète*)
 - la même clé sert à chiffrer et déchiffrer
 - les 2 parties doivent connaître la clé avant la communication
- Chiffrement **asymétrique**
 - 2 clés différentes :
 - une sert à chiffrer
 - une autre sert à déchiffrer
 - la clé publique peut (doit) être distribuer largement

clé **publique**
clé **privée**

Principe du chiffrement du canal

- Les 2 parties utilisent une **clé de session**
 - une clé pour un chiffrement **symétrique**
 - utilisée pour tous les échanges pendant la session
- L'échange de cette clé est fait par un chiffrement **assymétrique**
 - ① le serveur envoie sa clé publique au client
 - ② le client choisit une *clé pour la session*
 - ③ la clé est envoyée au serveur en utilisant la clé publique du serveur
 - ④ le serveur renvoie la clé de session choisie au client
 - ⑤ le serveur reçoit la clé de session choisie
 - ⑥ les deux parties utilisent la clé de session pour tous les échanges
- Il faut connaître la clé publique du serveur
 - elle est transmise sur le réseau *interception, modification potentielle*
 - **il faut vérifier son intégrité la première fois**
 - comparaison clé reçue et clé physiquement sur le serveur

Principe du chiffrement du canal

- Les 2 parties utilisent une **clé de session**
 - une clé pour un chiffrement **symétrique**
 - utilisée pour tous les échanges pendant la session
- L'échange de cette clé est fait par un chiffrement **assymétrique**
 - 1 le serveur envoie sa clé publique au client
 - 2 le client choisit une *clé pour la session*
 - 3 le client chiffre cette *clé de session* avec la clé publique du serveur
 - 4 le client envoie la *clé de session* chiffrée au serveur
 - 5 le serveur reçoit la *clé de session* chiffrée
 - 6 le serveur déchiffre la *clé de session* avec sa clé privée
- Il faut connaître la clé publique du serveur
 - elle est transmise sur le réseau *interception, modification potentielle*
 - **il faut vérifier son intégrité la première fois**
 - comparaison clé reçue et clé physiquement sur le serveur

Principe du chiffrement du canal

- Les 2 parties utilisent une **clé de session**
 - une clé pour un chiffrement **symétrique**
 - utilisée pour tous les échanges pendant la session
- L'échange de cette clé est fait par un chiffrement **assymétrique**
 - 1 le serveur envoie sa clé publique au client
 - 2 le client choisit une *clé pour la session*
 - 3 le client chiffre cette *clé de session* avec la clé publique du serveur
 - 4 le client envoie la *clé de session* chiffrée au serveur
 - 5 le serveur reçoit la *clé de session* chiffrée
 - 6 le serveur déchiffre la *clé de session* avec sa clé privée
- Il faut connaître la clé publique du serveur
 - elle est transmise sur le réseau *interception, modification potentielle*
 - **il faut vérifier son intégrité la première fois**
 - comparaison clé reçue et clé physiquement sur le serveur

Principe du chiffrement du canal

- Les 2 parties utilisent une **clé de session**
 - une clé pour un chiffrement **symétrique**
 - utilisée pour tous les échanges pendant la session
- L'échange de cette clé est fait par un chiffrement **assymétrique**
 - 1 le serveur envoie sa clé publique au client
 - 2 le client choisit une *clé pour la session*
 - 3 le client chiffre cette *clé de session* avec la clé publique du serveur
 - 4 le client envoie la *clé de session* chiffrée au serveur
 - 5 le serveur reçoit la *clé de session* chiffrée
 - 6 le serveur déchiffre la *clé de session* avec sa clé privée
- Il faut connaître la clé publique du serveur
 - elle est transmise sur le réseau *interception, modification potentielle*
 - **il faut vérifier son intégrité la première fois**
 - comparaison clé reçue et clé physiquement sur le serveur

Principe du chiffrement du canal

- Les 2 parties utilisent une **clé de session**
 - une clé pour un chiffrement **symétrique**
 - utilisée pour tous les échanges pendant la session
- L'échange de cette clé est fait par un chiffrement **assymétrique**
 - 1 le serveur envoie sa clé publique au client
 - 2 le client choisit une *clé pour la session*
 - 3 le client chiffre cette *clé de session* avec la clé publique du serveur
 - 4 le client envoie la *clé de session* chiffrée au serveur
 - 5 le serveur reçoit la *clé de session* chiffrée
 - 6 le serveur déchiffre la *clé de session* avec sa clé privée
- Il faut connaître la clé publique du serveur
 - elle est transmise sur le réseau *interception, modification potentielle*
 - **il faut vérifier son intégrité la première fois**
 - comparaison clé reçue et clé physiquement sur le serveur

Principe du chiffrement du canal

- Les 2 parties utilisent une **clé de session**
 - une clé pour un chiffrement **symétrique**
 - utilisée pour tous les échanges pendant la session
- L'échange de cette clé est fait par un chiffrement **assymétrique**
 - 1 le serveur envoie sa clé publique au client
 - 2 le client choisit une *clé pour la session*
 - 3 le client chiffre cette *clé de session* avec la clé publique du serveur
 - 4 le client envoie la *clé de session* chiffrée au serveur
 - 5 le serveur reçoit la *clé de session* chiffrée
 - 6 le serveur déchiffre la *clé de session* avec sa clé privée
- Il faut connaître la clé publique du serveur
 - elle est transmise sur le réseau *interception, modification potentielle*
 - **il faut vérifier son intégrité la première fois**
 - comparaison clé reçue et clé physiquement sur le serveur

Principe du chiffrement du canal

- Les 2 parties utilisent une **clé de session**
 - une clé pour un chiffrement **symétrique**
 - utilisée pour tous les échanges pendant la session
- L'échange de cette clé est fait par un chiffrement **assymétrique**
 - 1 le serveur envoie sa clé publique au client
 - 2 le client choisit une *clé pour la session*
 - 3 le client chiffre cette *clé de session* avec la clé publique du serveur
 - 4 le client envoie la *clé de session* chiffrée au serveur
 - 5 le serveur reçoit la *clé de session* chiffrée
 - 6 le serveur déchiffre la *clé de session* avec sa clé privée
- Il faut connaître la clé publique du serveur
 - elle est transmise sur le réseau *interception, modification potentielle*
 - **il faut vérifier son intégrité la première fois**
 - comparaison clé reçue et clé physiquement sur le serveur

Principe du chiffrement du canal

- Les 2 parties utilisent une **clé de session**
 - une clé pour un chiffrement **symétrique**
 - utilisée pour tous les échanges pendant la session
- L'échange de cette clé est fait par un chiffrement **assymétrique**
 - 1 le serveur envoie sa clé publique au client
 - 2 le client choisit une *clé pour la session*
 - 3 le client chiffre cette *clé de session* avec la clé publique du serveur
 - 4 le client envoie la *clé de session* chiffrée au serveur
 - 5 le serveur reçoit la *clé de session* chiffrée
 - 6 le serveur déchiffre la *clé de session* avec sa clé privée
- Il faut connaître la clé publique du serveur
 - elle est transmise sur le réseau *interception, modification potentielle*
 - **il faut vérifier son intégrité la première fois**
 - comparaison clé reçue et clé physiquement sur le serveur

Principe du chiffrement du canal

- Les 2 parties utilisent une **clé de session**
 - une clé pour un chiffrement **symétrique**
 - utilisée pour tous les échanges pendant la session
- L'échange de cette clé est fait par un chiffrement **assymétrique**
 - 1 le serveur envoie sa clé publique au client
 - 2 le client choisit une *clé pour la session*
 - 3 le client chiffre cette *clé de session* avec la clé publique du serveur
 - 4 le client envoie la *clé de session* chiffrée au serveur
 - 5 le serveur reçoit la *clé de session* chiffrée
 - 6 le serveur déchiffre la *clé de session* avec sa clé privée
- Il faut connaître la clé publique du serveur
 - elle est transmise sur le réseau *interception, modification* potentielle
 - **il faut vérifier son intégrité la première fois**
 - comparaison clé reçue et clé physiquement sur le serveur

OpenSSH : clés de serveur

Localisation

- clé privée du serveur `/etc/ssh/ssh_host_ecdsa_key`
- clé public du serveur `/etc/ssh/ssh_host_ecdsa_key.pub`

Manipulation

- afficher une empreinte (raccourci, *fingerprint*) de la clé publique

```
ssh-keygen -l -f /etc/ssh/ssh_host_ecdsa_key.pub
```

Authentification

Par mot de passe dans le canal chiffré

- méthode standard ...
 - comparaison des versions chiffrées des mots de passe
 - envoi du mot de passe entre les 2 machines (client vers serveur)
- ... mais *protégée* par le chiffrement de la session

Par échange de clés

- Méthode
 - 1 l'utilisateur génère une paire de clé asymétrique
 - 2 il donne sa clé publique à un utilisateur distant
 - 3 lors de la tentative de connection une **poignée de main cryptographique** est faite
- La clé privée est stockée dans un fichier
 - pour protéger son accès le fichier est chiffré symétriquement
 - clé de chiffrement du fichier = *passphrase*
 - utilisé à chaque utilisation de la clé privée
 - si elle est vide la clé n'est pas chiffrée

Authentification

Par mot de passe dans le canal chiffré

- méthode standard ...
 - comparaison des versions chiffrées des mots de passe
 - envoi du mot de passe entre les 2 machines (client vers serveur)
- ... mais *protégée* par le chiffrement de la session

Par échange de clés

- Méthode
 - 1 l'utilisateur génère une paire de clé asymétrique
 - 2 il donne sa clé publique à un utilisateur distant
 - 3 lors de la tentative de connection une **poignée de main cryptographique** est faite
- La clé privée est stockée dans un fichier
 - pour protéger son accès le fichier est chiffré symétriquement
 - clé de chiffrement du fichier = *passphrase*
 - utilisé à chaque utilisation de la clé privée
 - si elle est vide la clé n'est pas chiffrée

Outils OpenSSH

- commandes
 - `ssh(1)` : commande principale
 - `ssh-keygen(1)` : gestion des clés asymétriques
 - `ssh-copy-id(1)` : copie d'une clé publique sur un compte distant
 - `ssh-agent(1)` : agent d'authentification (conserve les clés en mémoire)
 - `ssh-add(1)` : manipulation des clés gérées par un agent
- configuration
 - client `ssh_config(5)`
 - `$HOME/.ssh/config` : configuration par défaut des programmes clients
 - `$HOME/.ssh/known_hosts` : clés publiques des serveurs connus
 - `$HOME/.ssh/authorized_keys` : clés publiques autorisées à se connecter
 - `$HOME/.ssh/id_rsa` : clé RSA privée
 - `$HOME/.ssh/id_rsa.pub` : clé RSA publique
 - serveur `sshd_config(5)`
 - `/etc/ssh/sshd_config` : configuration du serveur
 - `/etc/ssh/ssh_config` : configuration par défaut du comportement des clients

Utilisation

ssh(1)

- Accès shell distant (machine et utilisateur distant)
 - possibilité de passer des commandes en paramètres
 - accès shell complet (tubes, `stdin`, `stdout`, `stderr` utilisables)
 - beaucoup d'options dont :
 - rediriger les connexions d'un port local sur un port distant (et vice versa)
- Syntaxe pour identifier le *distant*

utilisateur @ machine

scp(1)

- Copie avec origine **ou** destination distante
 - quasiment la même syntaxe que la commande `cp`
- Syntaxe pour identifier le *distant*

utilisateur @ machine : chemin

sftp(1)

- Transfert interactif de fichiers à distance
 - fonctionnement similaire à `ftp(1)` mais dans un canal chiffré
 - permet de manipuler les fichiers distants
- Syntaxe pour identifier le *distant* identique à `ssh(1)`

Utilisation

ssh(1)

- Accès shell distant (machine et utilisateur distant)
 - possibilité de passer des commandes en paramètres
 - accès shell complet (tubes, `stdin`, `stdout`, `stderr` utilisables)
 - beaucoup d'options dont :
 - rediriger les connexions d'un port local sur un port distant (et vice versa)
- Syntaxe pour identifier le *distant*

utilisateur @ machine

scp(1)

- Copie avec origine **ou** destination distante
 - quasiment la même syntaxe que la commande `cp`
- Syntaxe pour identifier le *distant*

utilisateur @ machine : chemin

sftp(1)

- Transfert interactif de fichiers à distance
 - fonctionnement similaire à `ftp(1)` mais dans un canal chiffré
 - permet de manipuler les fichiers distants
- Syntaxe pour identifier le *distant* identique à `ssh(1)`

Utilisation

ssh(1)

- Accès shell distant (machine et utilisateur distant)
 - possibilité de passer des commandes en paramètres
 - accès shell complet (tubes, `stdin`, `stdout`, `stderr` utilisables)
 - beaucoup d'options dont :
 - rediriger les connexions d'un port local sur un port distant (et vice versa)
- Syntaxe pour identifier le *distant*

utilisateur @ machine

scp(1)

- Copie avec origine **ou** destination distante
 - quasiment la même syntaxe que la commande `cp`
- Syntaxe pour identifier le *distant*

utilisateur @ machine : chemin

sftp(1)

- Transfert interactif de fichiers à distance
 - fonctionnement similaire à `ftp(1)` mais dans un canal chiffré
 - permet de manipuler les fichiers distants
- Syntaxe pour identifier le *distant* identique à `ssh(1)`