TP 4 Usages du chiffrement : protocole SSH et logiciel OpenSSH

Objectifs du TP

L'objectif de ce TP est de comprendre l'utilisation du protocole SSH pour sécuriser les échanges entre deux stations.

---

1 Secure Shell (SSH)

1.1 Introduction à SSH

SSH chiffre toutes les communications grâce à une **`clé de session`** entre un client SSH et un serveur SSH.

De plus, SSH effectue une double authentification :

1. le logiciel client authentifie la machine serveur, ceci afin de se prémunir contre un vol de l'adresse IP du serveur ou contre une réponse DNS forgée par un pirate (attaque *man in the middle*).
2. le logiciel serveur authentifie l'utilisateur client, qui doit prouver son identité.

La première authentification se fait par un algorithme à clé publique (RSA, DSA, ECDSA, ED25519[1]. La deuxième se fait soit par mot de passe, soit par algorithme à clé publique.

Deux versions du protocole SSH existent :

— SSH1 : connaît des faiblesses si bien que son utilisation n'est plus recommandée.

— SSH2 : est le protocole recommandé (obligatoire depuis la version 8.8 de OpenSSH)

`RSA` (*Rivest-Shamir-Adelman*), **`DSA`** (*Digital Signature Algorithm*), **`ECDSA`** (*Elliptic curve digital signature algorithm*) ou, **`ED25519`** (variante de *Edwards-curve Digital Signature Algorithm*) sont des algorithmes asymétriques. Lorsque ces algorithmes sont utilisés pour de l'authentification, une personne ou une machine voulant prouver son identité génère une paire de clés. Une des clés est dite "publique" et la personne peut la diffuser à tous ceux qui doivent vérifier son identité. L'autre clé est dite "privée" et c'est elle qui servira à la personne pour prouver son identité. Le protocole d'authentification est tel que si un tiers espionne les échanges entres les 2 parties, il ne n'obtient aucune information sur la clé privée. Il est donc incapable de s'authentifier à la place de la personne originale.

Les algorithmes à clé publique sont considérés comme les plus sûrs connues actuellement, mais ils posent le problème de la distribution de la clé publique. Si un pirate est capable de distribuer sa clé publique à la place de la clé publique d'une personne légitime, l'algorithme n'assure plus aucune sécurité.

SSH effectue une double authentification :

1. le logiciel client **authentifie la machine serveur**, ceci afin de se prémunir contre un vol de l'adresse IP du serveur ou contre une réponse DNS forgée par un pirate (attaque *man in the middle*). L'authentification se fait par l'algorithme RSA ou DSA .

1. la plupart des ces algorithmes ne sont pas resistants à une attaque quantique. Voir le rapport NIST 2016.

2. le logiciel serveur **authentifie l'utilisateur**, qui doit prouver son identité. L'authentification de l'utilisateur peut se faire de 2 façons :
   — Si l'utilisateur n'a pas de paire de clés : envoi du mot de passe par le client. Cet envoi est chiffré à l'aide de l'algorithme Diffie-Hellman par génération de clés.
   — Si l'utilisateur a une paire de clés, et si le serveur connaît la clé publique de l'utilisateur, celui-ci peut faire la preuve de son identité sans envoyer son mot de passe.

En résumé, le protocole utilise **trois types de clés** pour le chiffrement :
— La paire de clés (publique et privée) du serveur pour échanger la clé de session (clé secrète)
— La paire de clés de l'utilisateur pour authentifier l'utilisateur,
— des clés de session générées aléatoirement et renouvelées régulièrement au cours de la session.

1.2 Authentifier un serveur SSH

Pour authentifier un serveur SSH, une méthode simple consiste à récupérer une clé publique du serveur pendant la 1^{re} connexion. Toutefois, bien que cette façon d'utiliser SSH est la moins contraignante pour l'utilisateur, elle est aussi la moins sûre.

1. Sur votre station, renommer ou effacer votre fichier **`~/.ssh/known_hosts`**
2. Faire une connexion SSH vers transit
3. Lire et interpréter l'avertissement de SSH :

```
The authenticity of host 'transit' can't be established.

ECDSA key fingerprint is
SHA256:RKOrkItEwozCDKwBTg+NyzN1BXP6atlLjZ5gxtyq0eI.

Are you sure you want to continue connecting (yes/no)?
```

4. Vérifier que l'empreinte de la clé est bien la même que celle affichée sur cette page : https://www-info.iut2.univ-grenoble-alpes.fr/intranet/informations/cellule-info/acces-exterieur.php
5. Que pensez vous du niveau de sécurité offert par ce moyen de vérifier la clé publique du serveur ?
6. Répondre 'yes' et lire la réponse de SSH :

```
Warning: Permanently added 'transit,193.55.51.227' (ECDSA)
to the list of known hosts.
```

7. Sur votre station, visualiser le contenu du fichier **`~/.ssh/known_hosts.`**
8. Regardez les lignes ajoutées dans le fichier. Pour chaque ligne, à quoi correspondent les trois parties ? Aidez-vous du man de sshd accessible ici `https://man.openbsd.org/sshd.8`. Sous quelle forme l'identifiant de la machine transit est-il stocké ?
9. A quoi correspondent les deux lignes ajoutées ? Cela correspond-il à une ou plusieurs clés ?
10. Refaire une nouvelle connexion SSH vers transit. Pourquoi la confirmation d'ajout de la clé n'est-elle plus demandée ?

1.3 Détails du protocole SSH

1. Faire une connexion SSH vers transit avec option de debug niveau 1 : `ssh -v ...`
2. Quelle est la version locale de OpenSSH ?
3. Quelle est la version distante de OpenSSH ?
4. Faire une connexion SSH vers transit avec option de debug niveau 2 : `ssh -vv ...`
5. Notez les
   — protocoles d'échanges de clés (lignes qui suivent "local client KEXINIT proposal" et "peer server KEXINIT proposal")
   — algorithmes de chiffrement (lignes "ciphers ctos" et "ciphers stoc")
   — algorithmes MAC (lignes "MACs ctos" et "MACs stoc")
6. Dans ce contexte, en vous aidant de

   `man ssh_config`

   trouvez ce que signifie "MAC" ?
7. Que signifient "ctos" et "stoc" ?
8. Finalement, quelles combinaisons d'algorithmes sont utilisées ?

1.4 Authentifier un utilisateur avec une clé

La méthode suivante permet d'éviter à l'utilisateur le risque de se faire voler son mot de passe si le client est dirigé à son insu vers un serveur contrôlé par une personne malveillante.

1. Sur votre station, générer une paire de clés : commande **ssh-keygen**. Lire le manuel pour trouver l'option permettant de générer une paire de clés plus longue que la longueur par défaut (choisir au moins 4096 bits soit 512 octets).
   En effet, un principe de sécurité en cryptographie est d'utiliser le maximum de bits que permet un logiciel (dans les limites d'une performance acceptable).
   Quand SSH demande une *passphrase* pour chiffrer votre clé privée, la choisir assez longue (au moins 12 caractères).
2. Quels fichiers ont été générés dans le répertoire `~/.ssh/` ?
3. Vérifier leurs permissions Unix. Pourquoi seul le propriétaire a-t-il des droits ?
4. Visualiser la clé publique. Par déduction, quel encodage est utilisé ?
5. Mettre cette clé publique dans votre compte sur transit, dans le fichier `~/.ssh/authorized_keys`
6. Faire une connexion SSH vers transit. Pourquoi la commande ssh ne demande plus votre mot de passe habituel ?

1.5 Utilisation d'un agent d'authentification

Taper sa *passphrase* à chaque connexion SSH est assez long et pénible. Le logiciel ssh-agent permet de mémoriser les clés SSH à la place de l'utilisateur, pour lui éviter de retaper

sa *passphrase* à chaque connexion. Il se comporte comme un trousseau de clés.

1. Vérifier qu'un processus ssh-agent tourne sur votre station et sous votre identité. Si ce n'est pas le cas, démarrer le processus avec la commande
   eval 'ssh-agent -s'
2. Utiliser la commande ssh-add pour saisir la *passphrase* de votre clé SSH
3. Faire une connexion SSH vers transit. SSH demande-t-il la *passphrase* ? Et les fois suivantes, pourquoi n'est-elle plus demandée ?

1.6 Transferts de fichiers par SSH

SSH permet aussi de transférer des fichiers d'une façon similaire à FTP, mais de façon sécurisée.

— Créez un fichier texte à votre nom
`touch \`whoami\`.txt`
— Utiliser la commande **scp** pour transférer un fichier dans le répertoire /tmp/ de la machine de votre voisin

```
scp votre-fichier votre-login@pc-XX-0YY:/tmp/
```

.
— Vérifier sur votre machine que vous avez bien reçu le fichier de votre voisin dans votre répertoire /tmp/
— Refaite la même commande scp. Pourquoi ne vous demande-t-on pas de redonner votre mot de passe alors que vous n'avez pas configuré la machine avec une clé publique ?

1.7 Mise en place de clés sur un serveur SSH

Cette partie nécessite les machines virtuelles. La VM Client nécessite de configurer l'interface ens3 avec l'adresse IP 11.22.33.98 (voir TP précédent) :

```
ip a add 11.22.33.98/24 dev ens3
```

La méthode de connexion précédente implique un risque de sécurité, pris en acceptant la clé du serveur sans la vérifier. La vérification a posteriori intervient trop tard, après avoir donné son mot de passe, potentiellement à une personne malveillante.

La méthode suivante permet d'éviter ce risque de sécurité en récupérant la clé du serveur **avant** la 1^{re} connexion.

1. Sur la VM serveur, démarrer le service ssh :

   ```
   service ssh start
   ```

2. Sur la VM serveur, calculer l'empreinte de la clé SSH de la machine avec la commande

   ```
   ssh-keygen -lf /etc/ssh/ssh_host_ecdsa_key.pub
   ```

   Cette empreinte est censée être transmise par un moyen sûr (autre que le réseau) de l'administrateur du serveur aux utilisateurs. On suppose que c'est le cas ici.
3. Depuis la VM client (si nécessaire, supprimer au préalable le fichier **~/.ssh/known_hosts**), faire une connexion SSH depuis le compte etu vers la VM serveur d'adresse 11.22.33.1
4. Vérifier que l'empreinte de la clé du serveur SSH sur la VM client est bien la bonne.