



SAE11 - Lot B : Accès SSH

Bleu → Florian

Violet → Victoria

Rouge → Raphaël

Orange → Ho Koloina

Vert → Thomas

I - Présentation du protocole SSH

SSH est un protocole de communication chargé d'établir une connexion chiffrée entre deux machines. Son implémentation principale se retrouve sous la suite d'outils OpenSSH.

Ce protocole est utile puisqu'il permet d'administrer des machines à distance sans accès physique requis après configuration.

Contrairement à Telnet, SSH est un protocole chiffrant la communication.

1 - Capacités

Les caractéristiques de SSH sont :

- Envoyer des commandes
- Transférer des fichiers à l'aide de SCP
- Mise en tunnel
 - Méthode pour transporter des données sur un réseau sans se soucier des restrictions de ce dernier
 - Établir des connexions efficaces et sécurisées entre deux réseaux

2 - Principe de fonctionnement

Nous allons ensuite voir le principe de fonctionnement de ssh, donc

Notion de client et serveur

SSH effectue une connexion entre un client et un serveur pour éviter d'avoir à y accéder physiquement.

Le client est celui qui se connecte à la machine hébergeant le serveur SSH.

Cependant, ne pas s'assurer de l'authenticité du serveur peut avoir de nombreux impacts sur la sécurité :

- risque d'usurpation (impossibilité de vérifier que l'on communique bien avec le bon serveur)
- exposition aux attaques de "l'homme du milieu", qui permet de récupérer l'ensemble des données du flux (frappes clavier, mots de passe, fichiers etc...).

Sous-protocoles

Le fonctionnement du protocole SSH repose sur 3 sous-protocoles :

- SSH-USERAUTH (authentification de la partie client)
C'est un peu un vigile : on doit lui montrer notre carte d'identité, lui donner un mot de passe et/ou encore lui donner une clé pour rentrer dans la boîte
- SSH-TRANS (transfert sécurisé → confidentialité et intégrité)
C'est comme si on mettait un cadenas sur l'enveloppe qu'on envoie vers la boîte.
- SSH-CONNECT (connexion qui permet le multiplexage de canaux de communication)
C'est comme si notre boîte avait pleins de petits tiroirs. Et c'est grâce à ces tiroirs que SSH peut envoyer plusieurs informations en même temps.

II - Sécurité de connexion

1 - Méthodes d'authentification

- Lors de la première connexion, on se connecte au serveur grâce à un utilisateur présent sur ce dernier et son mot de passe : `ssh "UTILISATEUR_SERVEUR"@"IP_SERVEUR"`.

En effet, l'authentification se fait par rapport à la connexion précédente, grâce à la clé fingerprint enregistrée.

- Enfin, SSH nous demande si on est certain d'initier la connexion puisqu'on ne s'est jamais connecté auparavant :

The authenticity of host "HOST_IP" can't be established.
RSA key fingerprint is "RSA_KEY_FINGERPRINT".

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

- De plus, au lieu d'utiliser le mot de passe de l'utilisateur, il est possible de mettre en place un système d'authentification par clé privée / publique :
 - Le client génère une paire de clé selon l'algorithme de son choix (il est expliqué par la suite, les algorithmes recommandés par l'ANSSI) avec `ssh-keygen`.
 - Argument `-t` : permet de choisir l'algorithme à utiliser
 - Argument `-b` : permet de choisir sa taille
 - Ex : création d'une paire de clés ECDSA en 256 bits
 - `ssh-keygen -t ecdsa -b 256`
 - Le client copie ensuite sa clé publique vers le serveur afin que le serveur sache qu'il faut autoriser chaque connexion venant du client avec `ssh-copy-id "EMPLACEMENT_CLÉ"`.

2 - Types de chiffrement et leurs rôles

L'entièreté de la connexion est chiffrée de manière symétrique. Ce n'est que lors de la phase d'authentification que le chiffrement asymétrique rentre en jeu.

Chiffrement symétrique

Dans le procédé de cryptographie symétrique, l'expéditeur et le destinataire se partagent une clé qui va servir au chiffrement et au déchiffrement du message.

Elle permet de chiffrer de façon rapide des volumes importants de données mais soulève le problème de l'échange des clés secrètes. (La transmission de la clé doit être confidentielle et la distribution des clés ne peut être utilisée à large échelle.)

Chiffrement asymétrique

Quand à lui, le chiffrement asymétrique est basée sur l'utilisation de 2 clés, une clé publique et une clé privée qui sont dépendantes l'une de l'autre donc la clé de chiffrement est différente de la clé de déchiffrement.

- La **clé publique** : elle est transmise à la personne avec laquelle on veut communiquer, elle permet de chiffrer le message.
- La **clé privée** : est une clé qui permet, quant à elle, de déchiffrer le message. Il faut la garder "précieusement".

Le chiffrement asymétrique permet donc d'échanger efficacement un petit volume de données, mais de façon sûre avec un interlocuteur.

III - Renforcement de la sécurité de la connexion

1 - Changement du port d'écoute

Le changement du port d'écoute par défaut (port 22) permet d'éviter une partie des menaces en obligeant les personnes voulant se connecter au serveur à avoir connaissance du port défini

alternativement.

1. Ouvrir le fichier de configuration `sshd_config` avec l'éditeur de texte de votre choix (on choisira ici nano)

```
sudo nano /etc/ssh/sshd_config
```

2. Changer la variable `Port` par la valeur choisie. (Pour éviter tout conflit avec des ports réservés, choisir une valeur supérieure à 1024.)
3. Enregistrer les modifications et ferme le fichier
4. Redémarrer le service SSH

```
sudo systemctl restart sshd
```

2 - Algorithmes

Le type d'algorithme utilisé est crucial puisqu'il va définir le niveau de sécurité de la communication SSH.

Types d'algorithmes de chiffrement asymétrique

- RSA (Rivest-Shamir-Adleman)
 - Taille min recommandée par l'ANSSI 2048 bits
- ECDSA & EdDSA (Elliptic Curve Digital Signature Algorithm)
 - Taille min recommandée par l'ANSSI 256 bits
- DSA (Digital Signature Algorithm) → non recommandé
 - Taille min 1024 bits

Recommandations

L'ANSSI recommande l'utilisation de l'algorithme ECDSA car utilise des clés plus petites avec une sécurité équivalente par rapport au RSA, qui utilise des clés beaucoup plus grande.

ECDSA est donc plus rapide, consomme moins de ressources, nécessite moins de bande passante et est généralement plus résistant aux attaques.

3 - Blocage au bout de plusieurs tentatives

Implémentation de base

Il est aussi possible de mettre en place un nombre limité de tentatives de connexions afin d'éviter les attaques d'authentification par brute force. Pour se faire il suffit de modifier la valeur de la variable `MaxTries` dans le fichier de configuration `sshd_config`.

À savoir que la valeur par défaut est définie sur 6.

Fail2Ban

Fail2Ban est un outils permettant de bannir une IP client après plusieurs tentatives de connexion échouées (il "surveille" le port SSH).

```
1) sudo apt install fail2ban

2) sudo apt install ssh

3) sudo systemctl start ssh

4) sudo systemctl status ssh

5) sudo systemctl start fail2ban

6) sudo systemctl status fail2ban

# On télécharge et vérifie que tous les programmes sont disponibles

7) cd /etc/fail2ban

8) ls

9) sudo nano jail.local

[sshd]
ignoreip = (mon IP)
backend = systemd
# backend = le moteur de surveillance des logs
enabled = true
port = ssh
# les ports à bloquer
filter = sshd
logpath = /var/log/auth.log
# logpath = emplacement des fichiers de log à surveiller
maxretry = 3
# Nombre d'essais de mdp avant d'être ban
bantime = 1m

# Ici on a "créé" la prison

10) sudo systemctl restart fail2ban

# On le restart pour appliquer les paramètres qu'on vient de faire

11) sudo systemctl enable fail2ban

12) sudo systemctl status fail2ban

# On vérifie que tout est bien disponible

13) ip a (sur la machine qui possède le fail2ban)
```

```
14) sudo grep "ban" var/leg/fail2ban.log
```

On cherche le mot "ban" dans le fichier qui répertorie les personnes interdites d'accès ?

Pour déban/ le sortir de la "prison" :

1) sudo fail2ban-client status

#On observe le nombre de "jail" c'est à dire de prison et la liste des "emprisonnés"

2) sudo fail2ban-client set <nom utilisateur>unban ip <IP>

Fail2Ban n'est pas un outil de sécurisation absolu du service. Il va juste rendre les attaques par forces brutes beaucoup plus longues. Si l'accès SSH n'est pas suffisamment sécurisé (avec un mot de passe trop faible par exemple), Fail2Ban n'empêchera pas l'attaquant de s'y connecter.

Ses objectifs sont :

- Éviter de surcharger les logs du système avec des milliers de tentatives de connexion
- Limiter la portée des attaques répétées provenant d'une même machine

Google Slide :

[https://docs.google.com/presentation/d/1ZHdXjxKmNXNaHFB25EkZgw54qRzdrxKjLI6cZNwiie0/edit?
usp=sharing](https://docs.google.com/presentation/d/1ZHdXjxKmNXNaHFB25EkZgw54qRzdrxKjLI6cZNwiie0/edit?usp=sharing)