Sécurisation Avancée de SSH : De l'Attaque à l'Authentification Asymétrique

Comprendre les vulnérabilités pour mieux fortifier l'accès serveur

L'Attaque (Hydra)

La Protection Active (Fail2ban)

Le Verrouillage (Clés SSH)

Le Mécanisme (Diagramme de Séquence)

Objectif : Rendre le Brute Force Impossible

Phase 1 : Red Team

Comprendre la faiblesse des mots de passe avec Hydra.

Phase 2 : Blue Team

Automatiser la défense avec Fail2ban.

Phase 3 : Hardening

Supprimer le vecteur d'attaque mot de passe via l'Authentification par Clé.

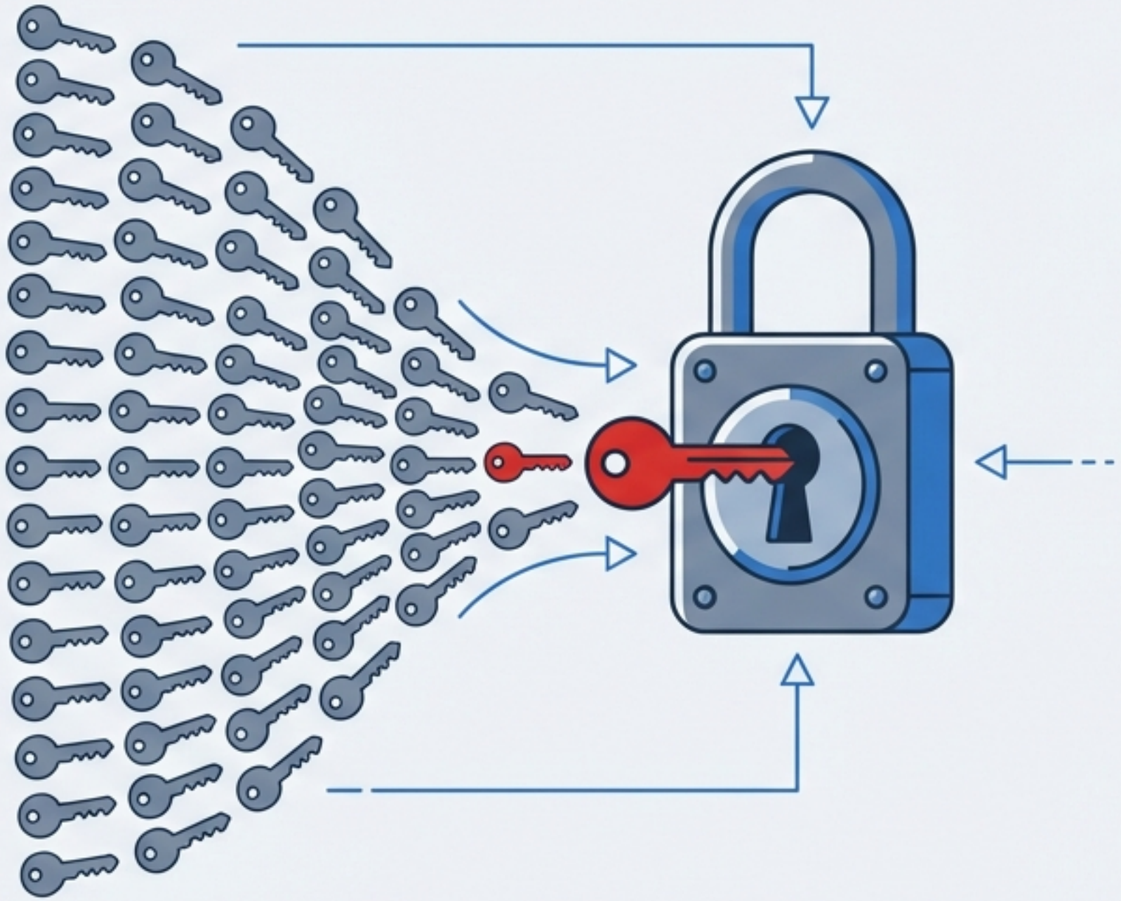
Context : Le port 22 (SSH) est la porte d'entrée principale des serveurs Linux. C'est la cible privilégiée des bots et des attaquants.

La Menace : Le 'Brute Force' sur le Port 22

Définition : Une tentative systématique de deviner le mot de passe en essayant toutes les combinaisons possibles.

L'Outil : THC Hydra – Un outil rapide et flexible pour l'audit de connexion réseau.

Concept Clé : Si votre mot de passe est dans un dictionnaire (comme pass.txt), Hydra le trouvera en quelques secondes.



L'Arme de l'Attaquant : Syntaxe Hydra

```
hydra [Utilisateur] [Mot de passe] [Cible] [Protocole]

  -l [nom] ———————— Teste un seul utilisateur (ex: victim).
  -P [file] ———————— Utilise un dictionnaire de mots de passe (ex: pass.txt).
  -t [nb] ———————— Nombre de connexions parallèles (Défaut 16, Conseillé 4).
  -f ———————————— Arrêt immédiat dès qu'un mot de passe est trouvé.
  -V ———————————— Mode verbeux (affiche les tentatives en temps réel).
```

Scénarios d'Attaque (Lab)

Scenario A : Cible Précise

```
hydra -l victim -P pass.txt ssh://[IP_CIBLE] -t 4 -f -V
```

Contexte : On connait l'utilisateur (victim), on cherche le mot de passe.

Scenario B : Spraying

```
hydra -L users.txt -P pass.txt ssh://[IP_CIBLE] -t 4
```

Contexte : On teste une liste d'utilisateurs courants (root, admin, user).

⚠️ **Ces tests doivent être réalisés uniquement dans un environnement autorisé (Lab/VM).**

La Vue du Serveur : Analyse des Logs

Fichier : /var/log/auth.log

```
Oct 10 14:32:01 server systemd: pam_unix(systemd-user:session): session opened for user nobody by (uid=0)
Oct 10 14:32:05 server sshd[3456]: Accepted publickey for admin from 192.168.1.10 port 54319 ssh2
Oct 10 14:32:15 server sshd[3460]: Invalid user support from 192.168.1.50
\033[1;41mFailed password for victim from 192.168.1.50 port 54321 ssh2
\nConnection closed by authenticating user 192.168.1.50
\nFailed password for victim from 192.168.1.50 port 54322 ssh2\033[0m
Oct 10 14:32:17 server sshd[3468]: Invalid user support from 192.168.1.50
\033[1;41mFailed password for victim from 192.168.1.50 port 54323 ssh2
\nConnection closed by authenticating user 192.168.1.50
\nFailed password for victim from 192.168.1.50 port 54323 ssh2
\nConnection closed by authenticating user 192.168.1.50
\nFailed password for victim from 192.168.1.50 port 54324 ssh2\033[0m
Oct 10 14:32:19 server sshd[3472]: Invalid user support from 192.168.1.50
\033[1;41mFailed password for victim from 192.168.1.50 port 54325 ssh2
\nConnection closed by authenticating user 192.168.1.50
\nFailed password for victim from 192.168.1.50 port 54326 ssh2\033[0m
Oct 10 14:32:21 server sshd[3480]: ...
```

Sans protection, ces lignes peuvent apparaître des milliers de fois par minute.

Inter Tight

La Protection Active : Fail2ban

1. Détecte les motifs d'échec répétés (Regex).

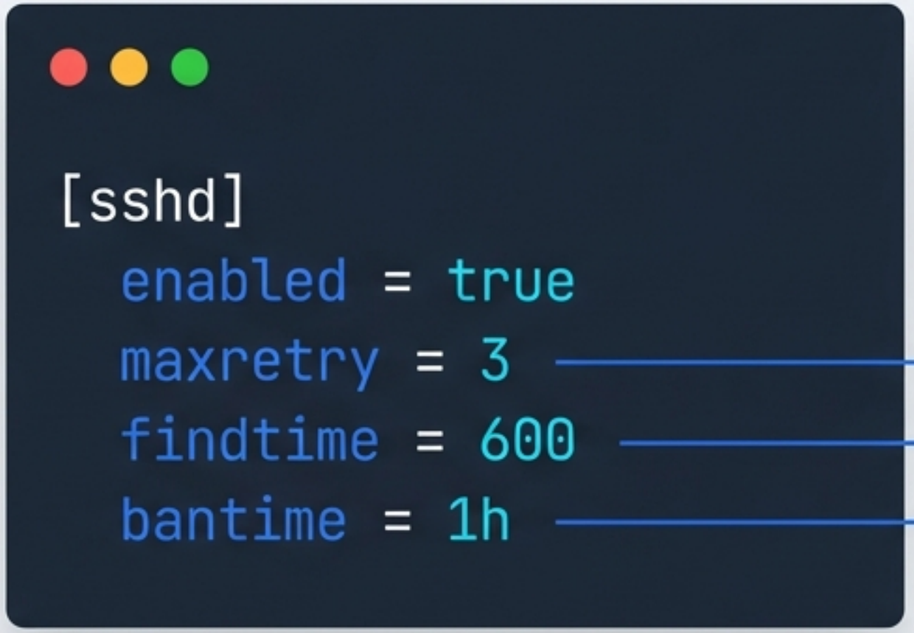
2. Bannit l'IP de l'attaquant.

```
sudo apt update && sudo apt install fail2ban -y
```

Analogie : Un vigile qui met sur liste noire toute personne qui se trompe de code 3 fois de suite.

Configuration de la 'Jail' (Prison)

Fichier : /etc/fail2ban/jail.local



- Nombre d'échecs autorisés avant ban.
- Fenêtre de temps (10 min) pour comptabiliser les échecs.
- Durée du bannissement de l'IP.

Vérification et Efficacité

`sudo fail2ban-client status sshd`

```
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  `- Total failed: 3
`- Actions
   |- Currently banned: 1
   `- Banned IP list: 192.168.1.50 ✅
```

Fail2ban mitige l'attaque, mais le vecteur d'authentification par mot de passe reste actif. Il faut changer la serrure.

Le Verrouillage : Authentification Asymétrique

Remplacer le mot de passe (symétrique/connu) par un défi mathématique cryptographique.

Clé Privée (id_rsa)

L'identité de l'utilisateur.
À garder strictement secrète.
Ne quitte jamais la machine client.

Clé Publique (id_rsa.pub)

Le cadenas.
À copier sur tous les serveurs cibles.
Peut être partagée sans risque.

Mise en Place : Génération et Déploiement

1. Génération (Client)

```
ssh-keygen -t rsa -b 4096
```

Note: Une clé de 4096 bits offre une résistance cryptographique élevée.

2. Déploiement (Vers la Cible)


```
ssh-copy-id victim@[IP_CIBLE]
```

Copie le contenu de **id_rsa.pub** dans le fichier **~/.ssh/authorized_keys** du serveur.

Mise en Place : Désactiver les Mots de Passe

Fichier : /etc/ssh/sshd_config (Sur le serveur)

```
# Désactivation de l'authentification par mot de passe
**PasswordAuthentication no**
PermitRootLogin no
```

```
sudo systemctl restart ssh
```

Résultat : Le serveur rejette désormais toute tentative de connexion sans clé, rendant Hydra inutile.

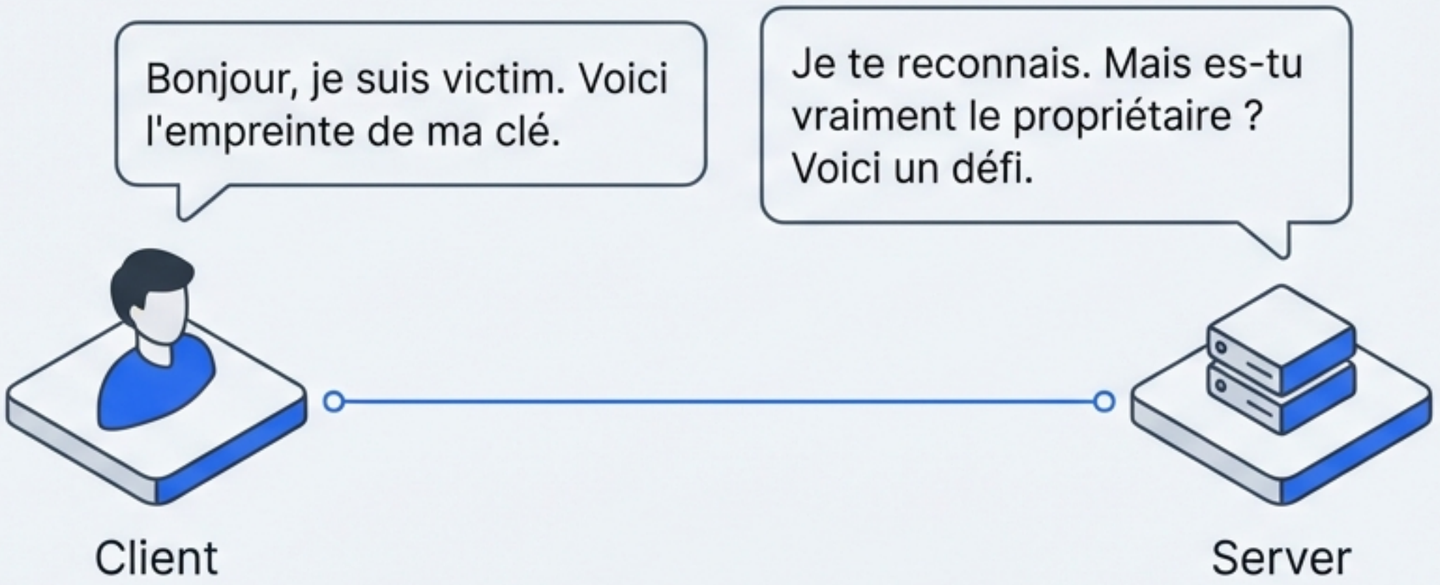
Le "Crash Test" Final

Action: Connexion SSH Légitime	✅ Succès immédiat sans mot de passe.
Action: Attaque Hydra	❌ Échec critique ("Password authentication not supported").
Action: Tentative Root	❌ Permission denied (Immédiat).

Conclusion : La surface d'attaque est drastiquement réduite.

Sous le Capot : Le Dialogue Invisible

Que se passe-t-il exactement quand vous tapez 'ssh user@ip' ?



Analyse du Handshake en 4 Étapes

Étape 1 : L'Identification

Action : Le client initie la connexion et envoie l'ID (Key ID) de sa paire de clés.

Vérification : Le serveur regarde dans ~/.ssh/authorized_keys.

Logique : Si trouvé → Passage au Challenge.

Étape 2 : Le Challenge (Le Défi)

Action : Le serveur génère un nombre aléatoire (Nonce).

Chiffrement : Le serveur chiffre ce message avec la Clé Publique de l'utilisateur.

Transmission : Le message chiffré est envoyé au client.

Étape 3 : La Preuve (Déchiffrement)

Action : Le client reçoit le 'blob' chiffré.

Déchiffrement : Le client utilise sa Clé Privée pour révéler le message aléatoire.

Réponse : Le client renvoie le message déchiffré au serveur.

Étape 4 : Validation et Accès

Vérification : Le serveur compare le message renvoyé par le client avec le message original.

Résultat : Match parfait = Identité prouvée = Accès autorisé.

Note : Le mot de passe n'a jamais traversé le réseau.

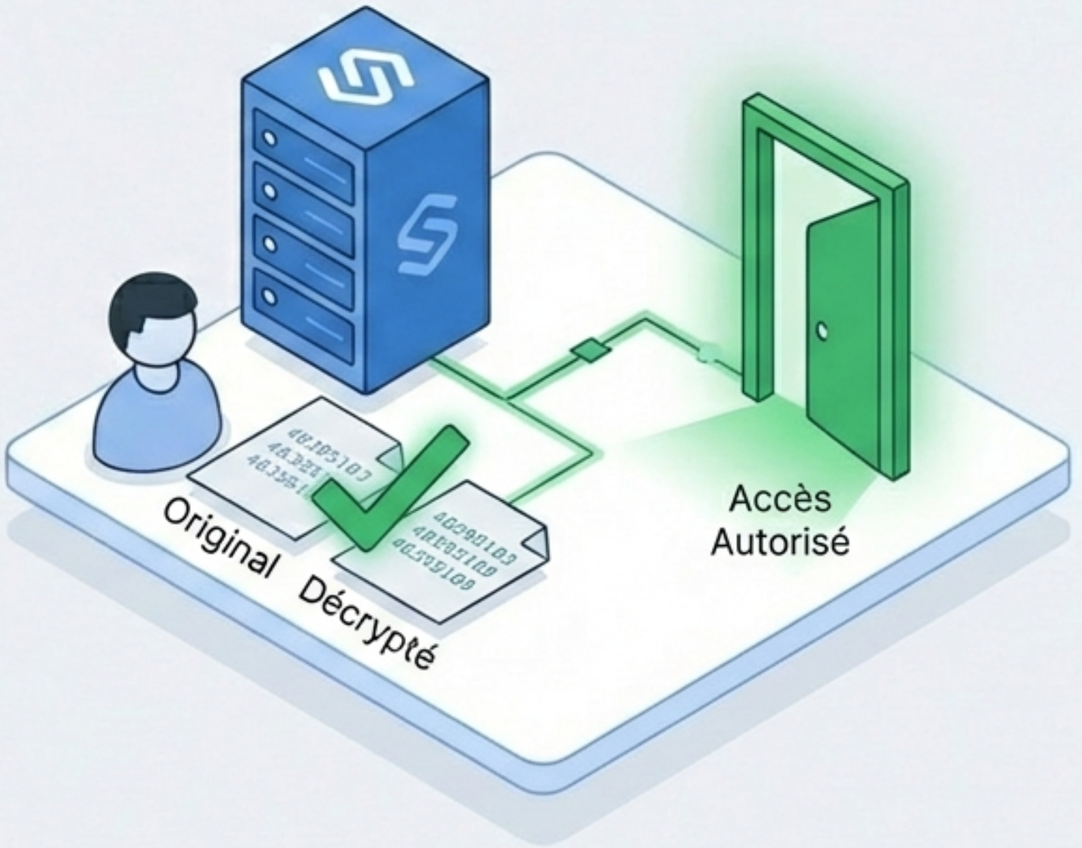
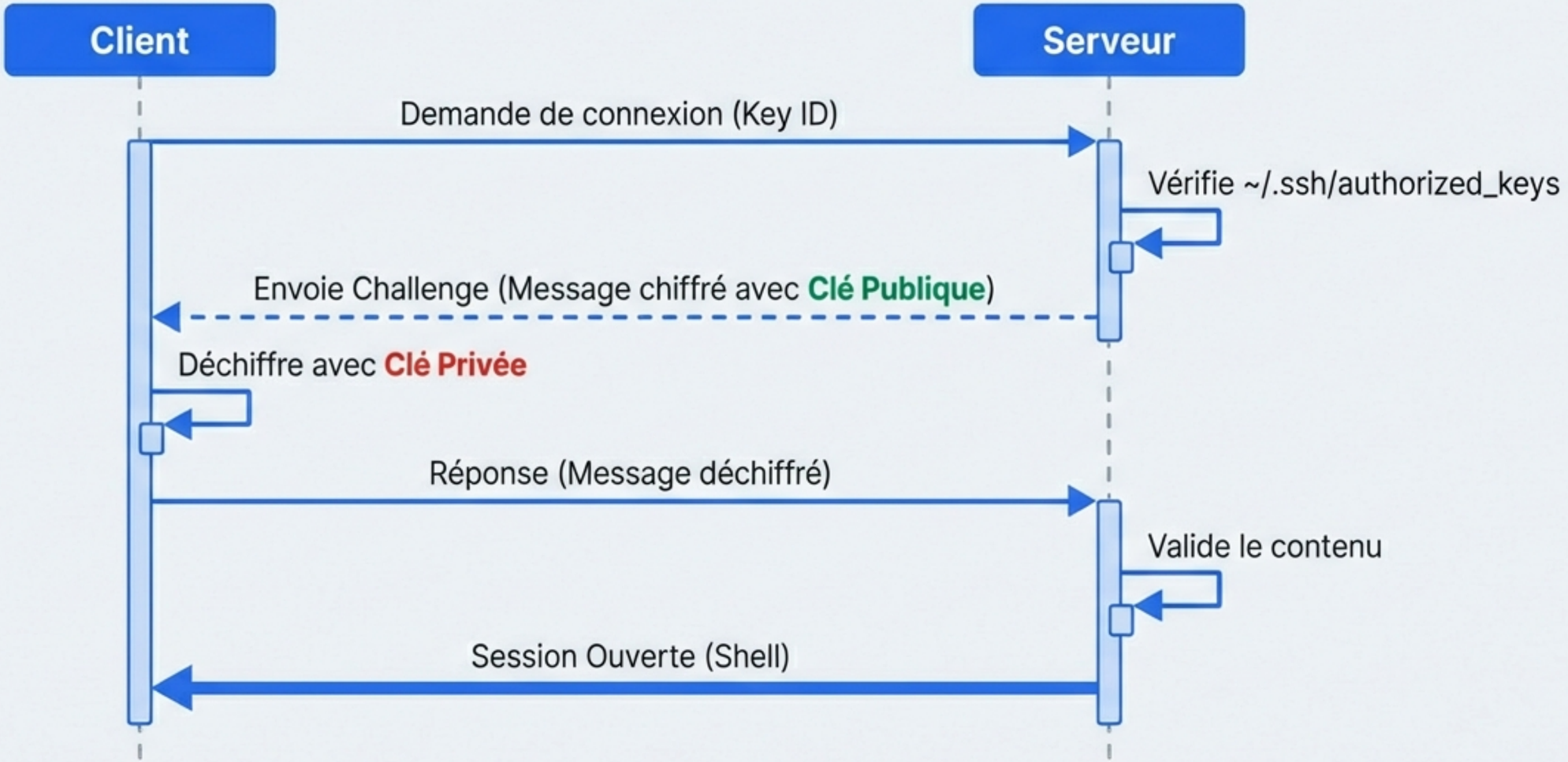


Diagramme de Séquence SSH Complet



Résumé : Une Défense en Profondeur

1. Prévention : Authentification par Clés (Supprime le risque de mot de passe faible).

2. Protection : Fail2ban (Bloque les scans et tentatives d'intrusion).

3. Compréhension : Maîtriser le handshake permet de mieux debugger et configurer.

"La sécurité n'est pas un produit, c'est un processus."