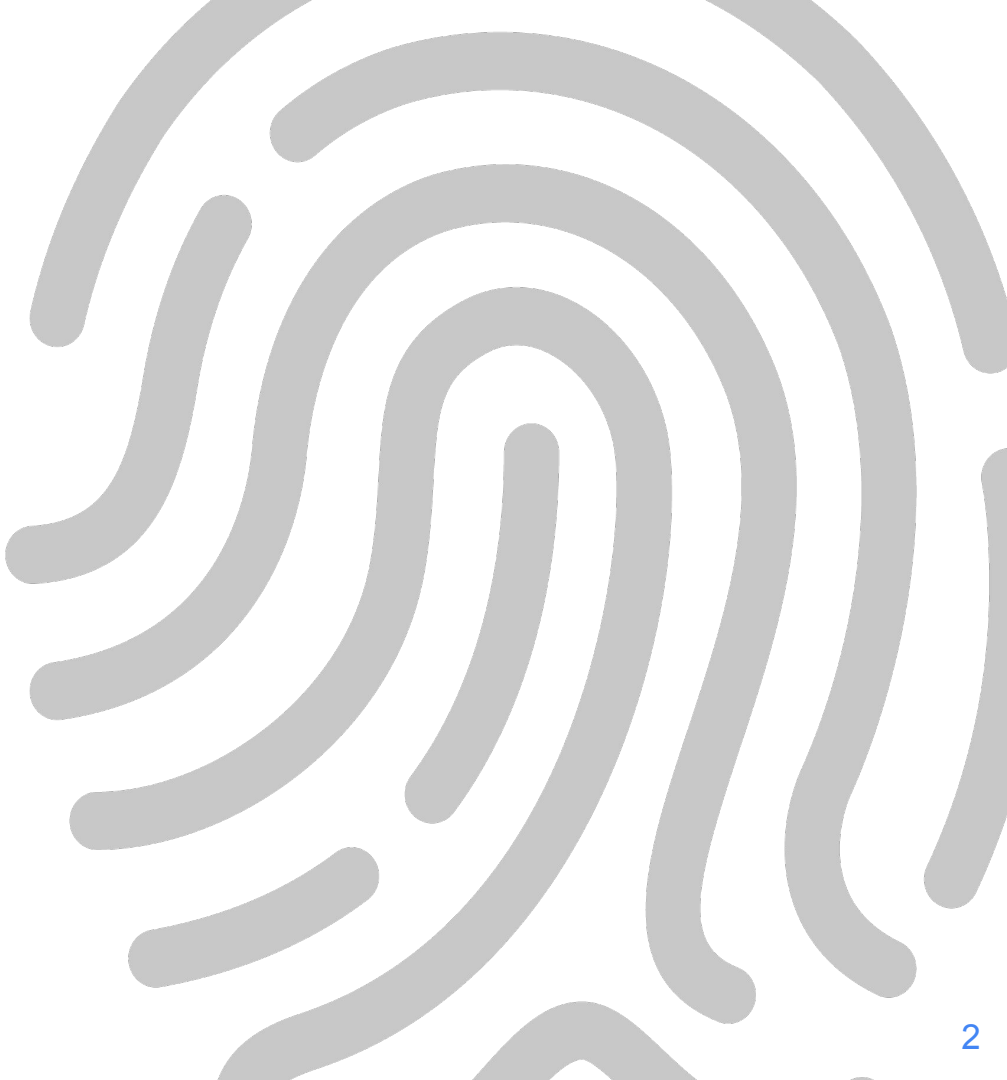
WILD CODE SCHOOL

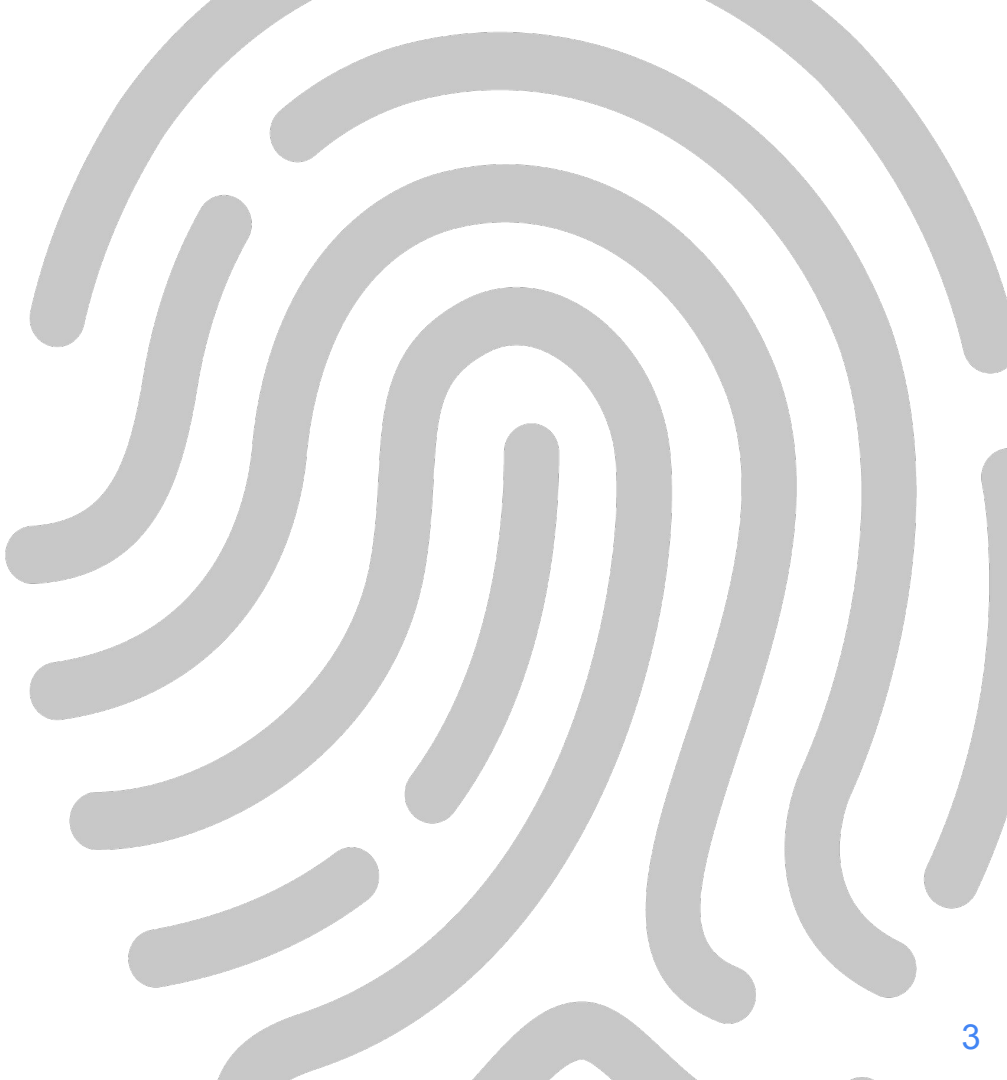
Secure Shell

SSH

- À quoi sert SSH ?
- Comment cela fonctionne ?



Plan



WILD CODE SCHOOL

Introduction

Historique

Premiers systèmes d'informations => émergence piles protocolaires visant l'échange de données entre machines : FTP, TELNET, RSH.

Protocoles non sécurisés :

- authentification
- intégrité
- confidentialité

Solution : SSH

Secure Shell (SSH)

- Protocole client serveur de communication sécurisé initié par Tatu Ylönen (RFC 4251, 4252, 4253 et 4254)
- Établissement d'une session de communication sécurisée entre 2 machines/terminaux => Assure confidentialité, authentification bidirectionnelle et intégrité
- Basé sur le modèle client / serveur
- Encapsulé dans **TCP** port standard **22** (par défaut)
- Utilisé sur Linux, Unix, Windows

Le protocole SSH

- Protocole de couche 7 (modèle OSI)
- Version 1 (obsolète) et version 2 (recommandée)
- 2023 SSH3 (alternative basée sur HTTP3 et QUIC)

Utilisation

- Ouverture de terminal et lancement de commande à distance
- Transfert de fichier :
  - **SFTP** : SSH File Transfer Protocol (Transfert sécurisé de fichiers)
  - **SCP** : Secure Copy (Copie de fichiers)
  - **SSHFS** : SSH File System (montage de dossier distant)
- Création de tunnel : Transfert de port (*port forwarding*), SOCKS…
- **X11 forwarding** : Ouverture de session graphique à distance

openSSH

Principale implémentation du protocole SSH

- Client (openssh-client) : ssh
- Serveur (openssh-server) : sshd
- Actuellement version 9.9p2 (juillet 2025)
- Maintenu par l'équipe openBSD

Connexion SSH

SSH permet l'utilisation de différents algorithmes cryptographiques.

Une connexion SSH commence donc par une négociation :

- Le client puis le serveur indique leurs versions de ssh et la version de leur implémentation
- Négociation des algorithmes utilisés

Connexion SSH (suite)

Chiffrement symétrique + HMAC de la communication =>

- Confidentialité : chiffrement symétrique et asymétrique
- Intégrité : Hachage

Authentification : Mot de passe, certificat, clé privée/publique.

Échange de clé de Diffie-Hellman (en pratique plutôt de type ECDH)
=> Confidentialité Persistante (PFS - Perfect Forward Secrecy)

À ce stade, la connexion devient confidentielle.

Cryptosystèmes

SSH distingue :

cipher :

chiffrements symétriques

cipher-auth :

chiffrements symétriques authentifié

kex :

échange de clés

key

types de clés

mac :

contrôles d'intégrité

```
wilder@host:~$ ssh -V
OpenSSH_8.9p1 Ubuntu-3, OpenSSL 3.0.2 15 Mar 2022
wilder@host:~$ ssh -Q cipher-auth
aes128-gcm@openssh.com
aes256-gcm@openssh.com
chacha20-poly1305@openssh.com
wilder@host:~$ ssh -Q key-plain
ssh-ed25519
sk-ssh-ed25519@openssh.com
ssh-rsa
ssh-dss
ecdsa-sha2-nistp256
ecdsa-sha2-nistp384
ecdsa-sha2-nistp521
sk-ecdsa-sha2-nistp256@openssh.com
```

WILD CODE SCHOOL

Authentification

Authentification SSH

SSH propose une authentification bidirectionnelle

- le client authentifie le serveur
- le serveur authentifie le client

L'authentification du serveur se déroule lors de l'initialisation de la connexion

L'authentification du client a lieu ensuite sur le canal établi avec le *bon* serveur

=> Confidentialité et intégrité des échanges avec un serveur authentifié

Authentification du serveur

Approche classique (et configuration par défaut)

=> Authentification par clés - Modèle TOFU (Trust On First Use)

Chaque serveur doit disposer d'une paires de clés (asymétriques) d'authentification.

À la première connexion à un serveur :

- Le serveur envoie sa clé publique
- Le client affiche l'empreinte de la clé et demande une validation

Authentification par certificats (moins courant) est aussi possible

- SSH permet une gestion de certificat avec une autorité interne et des certificats auto-signés (exemple de mise en place)

Clés serveur

Les clés du serveur ssh

- Dans **/etc/ssh/** (par défaut)
- **ssh_host_<algo>_key**
- **ssh_host_<algo>_key.pub**

À l'installation

- génération de clés pour l'ensemble des algorithmes recommandés

Affichage de l'empreinte

- **ssh-keygen -l**

```
wilder@server:~$ ls /etc/ssh/ssh_host_*
/etc/ssh/ssh_host_ecdsa_key         /etc/ssh/ssh_host_ed25519_key.pub
/etc/ssh/ssh_host_ecdsa_key.pub     /etc/ssh/ssh_host_rsa_key
/etc/ssh/ssh_host_ed25519_key       /etc/ssh/ssh_host_rsa_key.pub
wilder@server:~$ ssh-keygen -lf /etc/ssh/ssh_host_ed25519_key.pub
256 SHA256:kPFlNQn+PHJ+PMOcHe490TpKDjAIv7qmLM9XiZn3ahs root@server (ED25519)
```

Connexion (suite)

Première connexion :

- Affichage de l'empreinte
- Vérification manuelle (en théorie)
- Acceptation

=> clé(s) copiée(s) dans **.ssh/known_hosts**

- Client envoi un challenge au serveur

=> prouve que tu possèdes la clé privée associée

- echec au challenge

= fin de connexion

```
wilder@host:~$ ssh server
The authenticity of host 'server
(fd26:ba41:c8d6:0:a00:27ff:fea8:3fdf)' can't be
established.
ED25519 key fingerprint is
SHA256:kPFlNQn+PHJ+PMOcHe490TpKDjAIv7qm
LM9XiZn3ahs.
This key is not known by any other names
Are you sure you want to continue connecting
(yes/no/[fingerprint])? yes
Warning: Permanently added 'server' (ED25519) to
the list of known hosts.
wilder@server's password:
```

Connexion (suite)

Connexions suivantes :

- Vérification correspondance clé reçue et clé connue
- Challenge serveur

✅ => suite de la connexion
❌ => possible usurpation !

```
wilder@host:~$ ssh server
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
SHA256:wAlhyiiioksvb9WnJWon2sT7yCvL88llyest1wyWYz8.
Please contact your system administrator.
Add correct host key in /home/wilder/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/wilder/.ssh/known_hosts:13
  remove with:
  ssh-keygen -f "/home/wilder/.ssh/known_hosts" -R "server"
Host key for server has changed and you have requested strict checking.
Host key verification failed.
```

Authentification du client

À la suite de la connexion : authentification du client par le serveur

Plusieurs possibilités d'authentification du client :

- Par mot de passe (**password**), basée sur les comptes du système
- Par clé asymétrique (**publickey**), suppose une paire de clé sur le client pour l'utilisateur donnée et que la clé publique soit connue du serveur
- Basée sur l'hôte (**hostbased**) : par clés, mais pour tous les utilisateurs d'un hôte donné
- À l'aide d'outils tiers : PAM, Kerberos via GSSAPI

Ces différentes possibilités peuvent éventuellement s'additionner

Génération de clés

Sur le client :

- Pas de clés par défaut
- Clés utilisateur dans ~/.ssh
- id_<algo> et id_<algo>.pub

Génération avec ssh-keygen

- Passphrase :
  => chiffrement de la clé privée

```
wilder@host:~$ ssh-keygen -t ecdsa -b 256
Generating public/private ecdsa key pair.
Enter file in which to save the key
(/home/wilder/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in
/home/wilder/.ssh/id_ecdsa
Your public key has been saved in
/home/wilder/.ssh/id_ecdsa.pub
The key fingerprint is:
SHA256:fi1V+zFtqOar2bMIiFG8KogqycMEBpzvV4N+k
BT7z0E wilder@host
The key's randomart image is:
```

Copier ses clés

Configuration par défaut de sshd :

- **publickey** ou **password**

ssh-copy-id :

- copie une clé publique via ssh
- via une autre authentification

On peut ensuite envisager de désactiver l'authentification par mot de passe

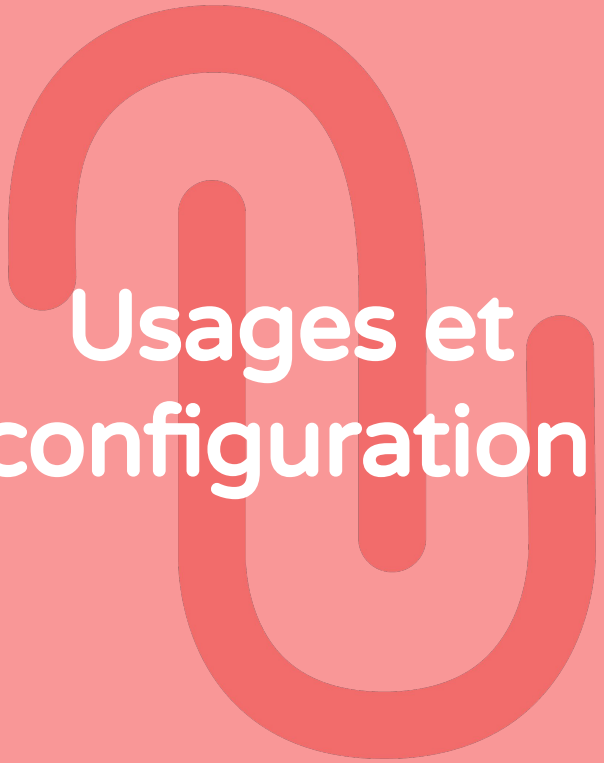
```
wilder@host:~$ ssh-copy-id server
/usr/bin/ssh-copy-id: INFO: attempting to log in with the
new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be
installed -- if you are prompted now it is to install the
new keys
wilder@server's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'server'"
and check to make sure that only the key(s) you
wanted were added.

wilder@host:~$ ssh server
Linux debian 5.10.0-15-amd64 #1 SMP Debian
5.10.120-1 (2022-06-09) x86_64
```

WILD CODE SCHOOL



Usages et configuration

Configuration du serveur

sshd : serveur ssh

- Gestion daemon classique (systemd ou init)
- Configuration : **/etc/ssh/sshd_config**
- Information sur la configuration : **man sshd_config**

Quelques idées de configuration

- Ne pas ouvrir sur le port 22 sur Internet (trop de bots)
  - Keyword: **Port xxx**
- Privilégier la connexion par clé, voir clé + mot de passe
  - Keyword: **AuthenticationMethods publickey,password**

Tester la configuration : **sshd -t** (évite une erreur de syntaxe)

voir **sshd -T** (test et affichage de la configuration)

Configuration du client

ssh : client ssh

- Configuration système globale : **/etc/ssh/ssh_config**
- Configuration particulière utilisateur : **~/.ssh/config** (prioritaire sur la conf globale)
- Information sur la configuration : **man ssh_config**
- Configurations peuvent être spécifiques à un serveur ou un cas de figure particulier

Quelques idées de configuration

- Vérifier le nom de domain et l'adresse IP (éviter des spoofing DNS)
  - Keyword: **CheckHostIP yes**
- Préciser le port par défaut
  - Keyword: **Port xxx**

Afficher la configuration : **ssh -G host** (pour host en particulier)

Shell distant

ssh : client ssh

- Serveur nécessaire (host ou IP)
- Format [user@]host ou
- ssh://[user@]host[:port] (Format URI)
- port par défaut : 22
- login par défaut : login local

Options courantes :

- p <port>
- l <login_name>
- v[v[v]] rendre ssh bavard

```
wilder@host:~$ ssh server
wilder@host:~$ ssh wilder@server
wilder@host:~$ ssh ssh://wilder@server:22
wilder@host:~$ ssh -l wilder -p 22 server
```

Copie de fichiers

scp : copie de fichier sécurisée

- Syntaxe : **scp source destination**

Avec source et destination

- un chemin (local)
- <[user@]host>:<chemin> (distant)

Copie source -> destination

Note : destination doit exister

Option : -r (copie de dossier)

```
wilder@host:~$ scp local_file server:distant_name
wilder@host:~$ scp wildo@server:/dir/distant_name
~/Downloads/
```

Pour quels usages la clé privée est chez moi ?

=> Génération de paire de clé sur hôte local

Accès à un serveur distant :

- Copie de la clé publique sur un serveur distant (dans le fichier **~/.ssh/authorized_keys** de l'utilisateur)

Gestion d'un dépôt distant sur Github avec Git :

- Clé publique ajoutée au compte GitHub
- Clé privée utilisée pour git push ou git pull
- opt. : création d'un fichier **~/.ssh/config**

Pour quels usages la clé privée est sur un serveur distant ?

Accès au parc d'un client via un serveur bastion :

- Connexion à un serveur bastion
- Clé publique et privée sur ce serveur bastion

Transfert de fichiers

sftp : transfert de fichiers interactif

- Syntaxe : **sftp destination**

Avec destination

- <[user@]host>:<chemin>
- sftp://[user@]host[:port][/chemin]

Ouverture d'une invite de commande

- Similaire à FTP
- commande help

Clients graphiques existent
(ex : FileZilla)

```
wilder@host:~$ sftp server
Connected to server.
sftp> help
Available commands:
bye                    Quit sftp
cd path                Change remote directory to 'path'
get [-afpR] remote [local]      Download file
lcd path               Change local directory to 'path'
ls [-1afhlnrSt] [path]      Display remote directory listing
put [-afpR] local [remote]      Upload file
```

Tunnels

ssh sert aussi de tunnel pour encapsuler du trafic et le transmettre de l'autre côté

- Redirection de port client -> serveur (**ssh -L port:host:hostport**)
  - Les paquets arrivant sur **client:port** transmis à **host:hostport** par le serveur
- Redirection de port serveur -> client (**ssh -R port:host:hostport**)
  - Les paquets arrivant sur **serveur:port** transmis à **host:hostport** par le client
- Proxy SOCKS (**ssh -D port**)
  - Le client SSH lance un serveur SOCKS
- Tunnels UNIX type point à point (**ssh -w local_tun_num:remote_tun_num**)
  - Création de pseudo device **tun** aux numéros spécifiés de chaque coté
- Interface graphique déportée - *X11 Forwarding* (**ssh -X**)
  - Permet le lancement d'application graphique à distance

Outils additionnels

L'installation de ssh apporte aussi des outils additionnels :

- **ssh-keygen** : gestion et génération de clés
- **ssh-agent** : stockage de clés privées en mémoire
- **ssh-add** : ajout de clés à ssh-agent
- **ssh-keyscan** : récupération de clés publiques sur des serveurs
- ***sftp-server*** : Sous système de gestion des connexions sftp
- ***ssh-keysign*** : Gestionnaire pour l'authentification *host-based*

WILD CODE SCHOOL

Fail2Ban

Fail2Ban

- Les attaques par force brute sont très courantes sur les serveurs SSH
- Possibilité de bloquer les adresses IP qui tentent de se connecter trop souvent avec un mauvais mot de passe
- Fail2Ban est un outil qui surveille les journaux système et bloque les adresses IP qui ont trop d'échecs de connexion
- Il peut être configuré pour surveiller n'importe quel service qui écrit des journaux de connexion (souvent utilisé pour SSH)

Plus d'informations sur https://doc.ubuntu-fr.org/fail2ban

Conclusion

- Aperçu du protocole SSH et de l'implémentation openSSH
- Mécanisme de connexion et d'authentification
- Différentes stratégie d'authentification du client et du serveur
- Différents usages de ssh

Sources

- Secure Shell sur WikipediA
- Les RFC 4251, 4252, 4253 et 4254
- Le site openssh officiel
- La SSH Academy sur ssh.com
- Les recommandations ANSSI pour la sécurisation d'openSSH