Version : **2026.01**

Dernière mise-à-jour : 2026/03/22 16:57

LDF517 - Gestion des Connexions à Distance

Contenu du Module

LAB #1 - Connexions à Distance

1.1 - Telnet

La commande **telnet** est utilisée pour établir une connexion à distance avec un serveur telnet :

```
  # telnet numero_ip
```

Important - Le service telnet revient à une redirection des canaux standards d'entrée et de sortie. Notez que la connexion n'est **pas** sécurisée. Pour fermer la connexion, il faut saisir la commande **exit**. La commande telnet n'offre pas de services de transfert de fichiers. Pour cela, il convient d'utiliser la command **ftp**.

Les options de cette commande sont :

```
root@debian11:~# which telnet
/usr/bin/telnet
root@debian11:~# telnet --help
telnet: invalid option -- '-'
Usage: telnet [-4] [-6] [-8] [-E] [-L] [-a] [-d] [-e char] [-l user]
        [-n tracefile] [ -b addr ] [-r] [host-name [port]]
```

3.2 - wget

La commande **wget** est utilisée pour récupérer un fichier via http, https ou ftp :

```
wget https://www.dropbox.com/scl/fi/c0cbo91y2i7qwjexeldgt/wget_file.txt?rlkey=g8fgje9z8oeqgb4nd2g7x3wkx
--2026-03-22 16:43:40--
```

```
https://www.dropbox.com/scl/fi/c0cbo91y2i7qwjexeldgt/wget_file.txt?rlkey=g8fgje9z8oeqgb4nd2g7x3wkx
Resolving www.dropbox.com (www.dropbox.com)... 162.125.11.18, 2620:100:6050:18::a27d:b12
Connecting to www.dropbox.com (www.dropbox.com)|162.125.11.18|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location:
https://uc47b1cd9dddd494a577d8544842.dl.dropboxusercontent.com/cd/0/inline/C9KstGml4v3e_uYkO7CO5vIAvBbyblAUwYvpHi
tDk6DRteAC7dL-L0LZzRQfBzzCvXOLo50aNZfIgy4smnZc3ZTb9ckSHae0BpYzaf2UbL_EtsjKzHGXOpCPkDnnwiuGkYwLhcDmGe-
XeMuqI1OtWSZp/file# [following]
--2026-03-22 16:43:41--
https://uc47b1cd9dddd494a577d8544842.dl.dropboxusercontent.com/cd/0/inline/C9KstGml4v3e_uYkO7CO5vIAvBbyblAUwYvpHi
tDk6DRteAC7dL-L0LZzRQfBzzCvXOLo50aNZfIgy4smnZc3ZTb9ckSHae0BpYzaf2UbL_EtsjKzHGXOpCPkDnnwiuGkYwLhcDmGe-
XeMuqI1OtWSZp/file
Resolving uc47b1cd9dddd494a577d8544842.dl.dropboxusercontent.com
(uc47b1cd9dddd494a577d8544842.dl.dropboxusercontent.com)... 162.125.11.15, 2620:100:6050:15::a27d:b0f
Connecting to uc47b1cd9dddd494a577d8544842.dl.dropboxusercontent.com
(uc47b1cd9dddd494a577d8544842.dl.dropboxusercontent.com)|162.125.11.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 46 [text/plain]
Saving to: ‘wget_file.txt?rlkey=g8fgje9z8oeqgb4nd2g7x3wkx’

wget_file.txt?rlkey=g 100%[======================>]      46  --.-KB/s    in 0s

2026-03-22 16:43:41 (61.1 MB/s) - ‘wget_file.txt?rlkey=g8fgje9z8oeqgb4nd2g7x3wkx’ saved [46/46]

root@debian11:~# cat wget_file.txt\?rlkey\=g8fgje9z8oeqgb4nd2g7x3wkx
This is a file retrieved by the wget command.
```

Les options de cette commande sont :

```
root@debian11:~# wget --help | more
GNU Wget 1.21, a non-interactive network retriever.
Usage: wget [OPTION]... [URL]...

Mandatory arguments to long options are mandatory for short options too.
```

```
Startup:
  -V,  --version                   display the version of Wget and exit
  -h,  --help                      print this help
  -b,  --background                go to background after startup
  -e,  --execute=COMMAND           execute a `.wgetrc'-style command

Logging and input file:
  -o,  --output-file=FILE          log messages to FILE
  -a,  --append-output=FILE        append messages to FILE
  -d,  --debug                     print lots of debugging information
  -q,  --quiet                     quiet (no output)
  -v,  --verbose                   be verbose (this is the default)
  -nv, --no-verbose                turn off verboseness, without being quiet
       --report-speed=TYPE         output bandwidth as TYPE.  TYPE can be bits
  -i,  --input-file=FILE           download URLs found in local or external FILE
  -F,  --force-html                treat input file as HTML
  -B,  --base=URL                  resolves HTML input-file links (-i -F)
                                     relative to URL
       --config=FILE               specify config file to use
       --no-config                 do not read any config file
       --rejected-log=FILE         log reasons for URL rejection to FILE

Download:
  -t,  --tries=NUMBER              set number of retries to NUMBER (0 unlimits)
       --retry-connrefused         retry even if connection is refused
       --retry-on-http-error=ERRORS    comma-separated list of HTTP errors to retry
  -O,  --output-document=FILE      write documents to FILE
  -nc, --no-clobber                skip downloads that would download to
                                     existing files (overwriting them)
       --no-netrc                  don't try to obtain credentials from .netrc
  -c,  --continue                  resume getting a partially-downloaded file
       --start-pos=OFFSET          start downloading from zero-based position OFFSET
       --progress=TYPE             select progress gauge type
       --show-progress             display the progress bar in any verbosity mode
```

```
  -N,  --timestamping              don't re-retrieve files unless newer than
                                     local
       --no-if-modified-since      don't use conditional if-modified-since get
                                     requests in timestamping mode
       --no-use-server-timestamps  don't set the local file's timestamp by
                                     the one on the server
  -S,  --server-response           print server response
       --spider                    don't download anything
  -T,  --timeout=SECONDS           set all timeout values to SECONDS
       --dns-timeout=SECS          set the DNS lookup timeout to SECS
       --connect-timeout=SECS      set the connect timeout to SECS
       --read-timeout=SECS         set the read timeout to SECS
  -w,  --wait=SECONDS              wait SECONDS between retrievals
                                     (applies if more then 1 URL is to be retrieved)
--More--
[q]
```

1.3 - ftp

Important - Si la commande **ftp** n'est pas installée sous Debian 11, installez-le à l'aide de la commande **apt install ftp** en tant que root.

La commande **ftp** est utilisée pour le transfert de fichiers. Une fois connecté, il convient d'utiliser la commande **help** pour afficher la liste des commandes disponibles :

```
ftp> help
Commands may be abbreviated.  Commands are:

!               dir             mdelete         qc              site
$               disconnect      mdir            sendport        size
account         exit            mget            put             status
```

```
append          form            mkdir           pwd             struct
ascii           get             mls             quit            system
bell            glob            mode            quote           sunique
binary          hash            modtime         recv            tenex
bye             help            mput            reget           tick
case            idle            newer           rstatus         trace
cd              image           nmap            rhelp           type
cdup            ipany           nlist           rename          user
chmod           ipv4            ntrans          reset           umask
close           ipv6            open            restart         verbose
cr              lcd             prompt          rmdir           ?
delete          ls              passive         runique
debug           macdef          proxy           send
ftp>
```

Le caractère **!** permet d'exécuter une commande sur la machine cliente

```
ftp> !pwd
/root
```

Pour transférer un fichier vers le serveur, il convient d'utiliser la commande **put** :

```
ftp> put nom_fichier_local nom_fichier_distant
```

Vous pouvez également transférer plusieurs fichiers à la fois grâce à la commande **mput**. Dans ce cas précis, il convient de saisir la commande suivante:

```
ftp> mput nom*.*
```

Pour transférer un fichier du serveur, il convient d'utiliser la commande **get** :

```
ftp> get nom_fichier
```

Vous pouvez également transférer plusieurs fichiers à la fois grâce à la commande **mget** (voir la commande **mput** ci-dessus).

Pour supprimer un fichier sur le serveur, il convient d'utiliser la commande **del** :

```
ftp> del nom_fichier
```

Pour fermer la session, il convient d'utiliser la commande **quit** :

```
ftp> quit
root@debian11:~#
```

1.4 - SSH

Présentation

La commande **ssh** est le successeur et la remplaçante de la commande **rlogin**. Il permet d'établir des connexions sécurisées avec une machine distante. SSH comporte cinq acteurs :

- Le **serveur SSH**
  - le démon sshd, qui s'occupe des authentifications et autorisations des clients,
- Le **client SSH**
  - ssh ou scp, qui assure la connexion et le dialogue avec le serveur,
- La **session** qui représente la connexion courante et qui commence juste après l'authentification réussie,
- Les **clefs**
  - **Couple de clef utilisateur asymétriques** et persistantes qui assurent l'identité d'un utilisateur et qui sont stockés sur disque dur,
  - **Clef hôte asymétrique et persistante** garantissant l'identité du serveur er qui est conservé sur disque dur
  - **Clef serveur asymétrique et temporaire** utilisée par le protocole SSH1 qui sert au chiffrement de la clé de session,
  - **Clef de session symétrique qui est générée aléatoirement** et qui permet le chiiffrement de la communication entre le client et le serveur. Elle est détruite en fin de session. SSH-1 utilise une seule clef tandis que SSH-2 utilise une clef par direction de la communication,
- La **base de données des hôtes connus** qui stocke les clés des connexions précédentes.

SSH fonctionne de la manière suivante pour la la mise en place d'un canal sécurisé:

- Le client contacte le serveur sur son port 22,
- Les client et le serveur échangent leur version de SSH. En cas de non-compatibilité de versions, l'un des deux met fin au processus,
- Le serveur SSH s'identifie auprès du client en lui fournissant :
  - Sa clé hôte,
  - Sa clé serveur,
  - Une séquence aléatoire de huit octets à inclure dans les futures réponses du client,
  - Une liste de méthodes de chiffrage, compression et authentification,
- Le client et le serveur produisent un identifiant identique, un haché MD5 long de 128 bits contenant la clé hôte, la clé serveur et la séquence aléatoire,
- Le client génère sa clé de session symétrique et la chiffre deux fois de suite, une fois avec la clé hôte du serveur et la deuxième fois avec la clé serveur. Le client envoie cette clé au serveur accompagnée de la séquence aléatoire et un choix d'algorithmes supportés,
- Le serveur déchiffre la clé de session,
- Le client et le serveur mettent en place le canal sécurisé.

SSH-1

SSH-1 utilise une paire de clefs de type RSA1. Il assure l'intégrité des données par une **Contrôle de Redondance Cyclique** (CRC) et est un bloc dit **monolithique**.

Afin de s'identifier, le client essaie chacune des six méthodes suivantes :

- **Kerberos**,
- **Rhosts**,
- **RhostsRSA**,
- Par **clef asymétrique**,
- **TIS**,
- Par **mot de passe**.

SSH-2

SSH-2 utilise **DSA** ou **RSA**. Il assure l'intégrité des données par l'algorithme **HMAC**. SSH-2 est organisé en trois **couches** :

- **SSH-TRANS** – Transport Layer Protocol,
- **SSH-AUTH** – Authentification Protocol,
- **SSH-CONN** – Connection Protocol.

SSH-2 diffère de SSH-1 essentiellement dans la phase authentification.

Trois méthodes d'authentification :

- Par **clef asymétrique**,
  - Identique à SSH-1 sauf avec l'algorithme DSA,
- **RhostsRSA**,
- Par **mot de passe**.

Les options de cette commande sont :

```
root@debian11:~# ssh --help
unknown option -- -
usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface]
           [-b bind_address] [-c cipher_spec] [-D [bind_address:]port]
           [-E log_file] [-e escape_char] [-F configfile] [-I pkcs11]
           [-i identity_file] [-J [user@]host[:port]] [-L address]
           [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port]
           [-Q query_option] [-R address] [-S ctl_path] [-W host:port]
           [-w local_tun[:remote_tun]] destination [command]
```

Authentification par mot de passe

L'utilisateur fournit un mot de passe au client ssh. Le client ssh le transmet de façon sécurisée au serveur ssh puis le serveur vérifie le mot de passe et l'accepte ou non.

Avantage:

- Aucune configuration de clef asymétrique n'est nécessaire.

Inconvénients:

- L'utilisateur doit fournir à chaque connexion un identifiant et un mot de passe,
- Moins sécurisé qu'un système par clef asymétrique.

Authentification par clef asymétrique

- Le **client** envoie au serveur une requête d'authentification par clé asymétrique qui contient le module de la clé à utiliser,
- Le **serveur** recherche une correspondance pour ce module dans le fichier des clés autorisés **~/.ssh/authorized_keys**,
  - Dans le cas où une correspondance n'est pas trouvée, le serveur met fin à la communication,
  - Dans le cas contraire le serveur génère une chaîne aléatoire de 256 bits appelée un **challenge** et la chiffre avec la **clé publique du client**,
- Le **client** reçoit le challenge et le décrypte avec la partie privée de sa clé. Il combine le challenge avec l'identifiant de session et chiffre le résultat. Ensuite il envoie le résultat chiffré au serveur.
- Le **serveur** génère le même haché et le compare avec celui reçu du client. Si les deux hachés sont identiques, l'authentification est réussie.

Configuration du Serveur

La configuration du serveur s'effectue dans le fichier **/etc/ssh/sshd_config** :

```
root@debian11:~# cat /etc/ssh/sshd_config
#       $OpenBSD: sshd_config,v 1.103 2018/04/09 20:41:22 tj Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.
```

```
Include /etc/ssh/sshd_config.d/*.conf

#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile     .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
```

```
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no

# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes
#GSSAPIStrictAcceptorCheck yes
#GSSAPIKeyExchange no

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the ChallengeResponseAuthentication and
# PasswordAuthentication.  Depending on your PAM configuration,
```

```
# PAM authentication via ChallengeResponseAuthentication may bypass
# the setting of "PermitRootLogin without-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and ChallengeResponseAuthentication to 'no'.
UsePAM yes

#AllowAgentForwarding yes
#AllowTcpForwarding yes
#GatewayPorts no
X11Forwarding yes
#X11DisplayOffset 10
#X11UseLocalhost yes
#PermitTTY yes
PrintMotd no
#PrintLastLog yes
#TCPKeepAlive yes
#PermitUserEnvironment no
#Compression delayed
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS no
#PidFile /var/run/sshd.pid
#MaxStartups 10:30:100
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none

# no default banner path
#Banner none

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*
```

```
# override default of no subsystems
Subsystem       sftp    /usr/lib/openssh/sftp-server

# Example of overriding settings on a per-user basis
#Match User anoncvs
#       X11Forwarding no
#       AllowTcpForwarding no
#       PermitTTY no
#       ForceCommand cvs server
```

Pour ôter les lignes de commentaires dans ce fichier, utilisez la commande suivante :

```
root@debian11:~# cd /tmp ; grep -E -v '^(#|$)'  /etc/ssh/sshd_config > sshd_config

root@debian11:/tmp# cat sshd_config
Include /etc/ssh/sshd_config.d/*.conf
PermitRootLogin yes
ChallengeResponseAuthentication no
UsePAM yes
X11Forwarding yes
PrintMotd no
AcceptEnv LANG LC_*
Subsystem       sftp    /usr/lib/openssh/sftp-server
```

Pour sécuriser le serveur ssh, ajoutez ou modifiez les directives suivantes :

```
AllowGroups adm
Banner /etc/issue.net
HostbasedAuthentication no
IgnoreRhosts yes
LoginGraceTime 60
LogLevel INFO
PermitEmptyPasswords no
PermitRootLogin no
```

```
PrintLastLog yes
Protocol 2
StrictModes yes
X11Forwarding no
```

Votre fichier ressemblera à celui-ci :

```
root@debian11:/tmp# vi sshd_config
root@debian11:/tmp# cat sshd_config
AllowGroups adm
Banner /etc/issue.net
HostbasedAuthentication no
IgnoreRhosts yes
LoginGraceTime 60
LogLevel INFO
PermitEmptyPasswords no
PermitRootLogin no
PrintLastLog yes
Protocol 2
StrictModes yes
X11Forwarding no
Include /etc/ssh/sshd_config.d/*.conf
ChallengeResponseAuthentication no
UsePAM yes
PrintMotd no
AcceptEnv LANG LC_*
Subsystem       sftp    /usr/lib/openssh/sftp-server
```

Renommez le fichier **/etc/ssh/sshd_config** en **/etc/ssh/sshd_config.old** :

```
root@debian11:/tmp# cp /etc/ssh/sshd_config /etc/ssh/sshd_config.old
```

Copiez le fichier **/tmp/sshd_config** vers **/etc/ssh/** :

```
root@debian11:/tmp# cp /tmp/sshd_config /etc/ssh
```

Redémarrez le service sshd :

```
root@debian11:/tmp# systemctl restart sshd

root@debian11:/tmp# systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2022-05-03 11:01:24 CEST; 7s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 4885 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 4888 (sshd)
      Tasks: 1 (limit: 4632)
     Memory: 1.1M
        CPU: 24ms
     CGroup: /system.slice/ssh.service
             └─4888 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

May 03 11:01:24 debian11.ittraining.loc sshd[4888]: Server listening on 0.0.0.0 port >
May 03 11:01:24 debian11.ittraining.loc systemd[1]: Stopping OpenBSD Secure Shell ser>
May 03 11:01:24 debian11.ittraining.loc sshd[4888]: Server listening on :: port 22.
May 03 11:01:24 debian11.ittraining.loc systemd[1]: ssh.service: Succeeded.
May 03 11:01:24 debian11.ittraining.loc systemd[1]: Stopped OpenBSD Secure Shell serv>
May 03 11:01:24 debian11.ittraining.loc systemd[1]: Starting OpenBSD Secure Shell ser>
May 03 11:01:24 debian11.ittraining.loc systemd[1]: Started OpenBSD Secure Shell serv>
lines 1-20/20 (END)
[q]
```

Mettez l'utilisateur **trainee** dans le groupe **adm** :

```
root@debian11:/tmp# groups trainee
trainee : trainee cdrom floppy audio dip src video plugdev netdev lpadmin scanner
```

```
root@debian11:/tmp# usermod -aG adm trainee
root@debian11:/tmp# groups trainee
trainee : trainee adm cdrom floppy audio dip src video plugdev netdev lpadmin scanner
```

Pour générer les clefs du serveur, saisissez la commande suivante en tant que **root**. Notez que la passphrase doit être **vide**.

```
root@debian11:/tmp# ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/root/.ssh/id_dsa): /etc/ssh/ssh_host_dsa_key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /etc/ssh/ssh_host_dsa_key
Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub
The key fingerprint is:
SHA256:emwF3Bq/2H+JGk5dNXgKBLTtbuC04byZmFJDxkUxso4 root@debian11.ittraining.loc
The key's randomart image is:
+---[DSA 1024]----+
|       .o*o.     |
|       .oo=   .  |
|      ..=..o . o.|
|      o+ =. . o..|
|     Eo.S+o. ..  |
|       ==+=o .   |
|      o *==o.. . |
|     . ooo=o. o  |
|      .o +o...   |
+----[SHA256]-----+

root@debian11:/tmp# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): /etc/ssh/ssh_host_rsa_key
/etc/ssh/ssh_host_rsa_key already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
```

```
Enter same passphrase again:
Your identification has been saved in /etc/ssh/ssh_host_rsa_key
Your public key has been saved in /etc/ssh/ssh_host_rsa_key.pub
The key fingerprint is:
SHA256:xrkpZ6lfF1hQIZEYJuH1L+Z2QGhQCf8Xwt9HPwTuT7Y root@debian11.ittraining.loc
The key's randomart image is:
+---[RSA 3072]----+
|      =+=+++oo   |
|     . *o+.o. .  |
|      . + = o. ..|
|       o + *.o...|
|        S * =..+o|
|       . * + .+.o|
|      . * + o  E |
|       = o o     |
|      ...        |
+----[SHA256]-----+

root@debian11:/tmp# ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/root/.ssh/id_ecdsa): /etc/ssh/ssh_host_ecdsa_key
/etc/ssh/ssh_host_ecdsa_key already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /etc/ssh/ssh_host_ecdsa_key
Your public key has been saved in /etc/ssh/ssh_host_ecdsa_key.pub
The key fingerprint is:
SHA256:38O9lqa1AHvviceNEbQ1AOUMspkYBpIpLlu/U05ymqo root@debian11.ittraining.loc
The key's randomart image is:
+---[ECDSA 256]---+
|   .o..o . ooo.  |
| . o. . o = + ...|
|. .    . +   + o.|
```

```
|...           o  |
|.o .    S .    . |
|.   o +  . = ..  |
|     X    o *.o= |
|    = .    . *B+.|
|E... .      o**  |
+----[SHA256]-----+

root@debian11:/tmp# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519): /etc/ssh/ssh_host_ed25519_key
/etc/ssh/ssh_host_ed25519_key already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /etc/ssh/ssh_host_ed25519_key
Your public key has been saved in /etc/ssh/ssh_host_ed25519_key.pub
The key fingerprint is:
SHA256:f8arQ5MBRGNJoj4eARYapvxf/MLxFFMZcKf1eLkgeow root@debian11.ittraining.loc
The key's randomart image is:
+--[ED25519 256]--+
|..+.  .+*o.+oo   |
|++ . . oo.o.+ o .|
|o.  o    + o o + |
|  .. ..   B . o .|
|   .+  +SE =   . |
|   ..oo =.=.     |
|    .. o +..+    |
|        . .o .   |
|          .o.    |
+----[SHA256]-----+
```

Les clefs publiques générées possèdent l'extension **.pub**. Les clefs privées n'ont pas d'extension :

```
root@debian11:/tmp# ls /etc/ssh
moduli         sshd_config.d          ssh_host_ecdsa_key         ssh_host_rsa_key
ssh_config     sshd_config.old        ssh_host_ecdsa_key.pub     ssh_host_rsa_key.pub
ssh_config.d   ssh_host_dsa_key       ssh_host_ed25519_key
sshd_config    ssh_host_dsa_key.pub   ssh_host_ed25519_key.pub
```

Re-démarrez ensuite le service sshd :

```
root@debian11:/tmp# systemctl restart sshd.service
root@debian11:/tmp# systemctl status sshd.service
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2022-05-03 11:30:09 CEST; 8s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 4942 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 4943 (sshd)
      Tasks: 1 (limit: 4632)
     Memory: 1.1M
        CPU: 24ms
     CGroup: /system.slice/ssh.service
             └─4943 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

May 03 11:30:09 debian11.ittraining.loc systemd[1]: Starting OpenBSD Secure Shell ser>
May 03 11:30:09 debian11.ittraining.loc sshd[4943]: Server listening on 0.0.0.0 port >
May 03 11:30:09 debian11.ittraining.loc sshd[4943]: Server listening on :: port 22.
May 03 11:30:09 debian11.ittraining.loc systemd[1]: Started OpenBSD Secure Shell serv>
lines 1-17/17 (END)
[q]
```

Configuration du Client

Saisissez maintenant les commandes suivantes en tant que **trainee** :

Important - Lors de la génération des clefs, la passphrase doit être **vide**.

```
root@debian11:/tmp# exit
logout
trainee@debian11:~$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/trainee/.ssh/id_dsa):
Created directory '/home/trainee/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/trainee/.ssh/id_dsa
Your public key has been saved in /home/trainee/.ssh/id_dsa.pub
The key fingerprint is:
SHA256:WijVQNwc9klBZxW4R8ZY5LwZK08/FaZCkWP0yKNj+20 trainee@debian11.ittraining.loc
The key's randomart image is:
+---[DSA 1024]----+
|     ooooo=++B=. |
|      .+oo.B*o+  |
|      . . +=.=+o |
|     . .  o o +=.|
|    . . S+ ..o= .|
|     . o. o .+ ..|
|      .  .    ...|
|          . .E  .|
|           ...   |
+----[SHA256]-----+

trainee@debian11:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/home/trainee/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/trainee/.ssh/id_rsa
Your public key has been saved in /home/trainee/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:r8id4Px97b9t7GMybe7OTaQU5lgaRjsTK/EAyMFdgjo trainee@debian11.ittraining.loc
The key's randomart image is:
+---[RSA 3072]----+
|     o.=oo+ o    |
|      = .. = +   |
|     .    . O +  |
|    E      o X . |
|     .  S   o o .|
|         .   . o |
|      .   .  .o..|
|     + + +  .oo*=|
|      =.= .. .XXB|
+----[SHA256]-----+

trainee@debian11:~$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/trainee/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/trainee/.ssh/id_ecdsa
Your public key has been saved in /home/trainee/.ssh/id_ecdsa.pub
The key fingerprint is:
SHA256:IL44+ZzExDxBFCt9nZtMgCQB/iuq9UtJC6uq/RhrPvg trainee@debian11.ittraining.loc
The key's randomart image is:
+---[ECDSA 256]---+
|..o+=o.          |
|.  +.. o .       |
| .. = o +        |
```

```
|  .= + + o       |
|  ..B   S        |
|   O.=           |
| oB.B            |
|o++@ .           |
|X+EoB.           |
+----[SHA256]-----+

trainee@debian11:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/trainee/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/trainee/.ssh/id_ed25519
Your public key has been saved in /home/trainee/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:yQ9mtIx1nK7D1vSZjkkbofXHZsTDp5P0rywiJIMX35M trainee@debian11.ittraining.loc
The key's randomart image is:
+--[ED25519 256]--+
|                 |
|         . .     |
|        o +      |
|      .* =   o   |
|     ..oS.=.  * .|
|    . ++oOE+ * * |
|     . +* =.= O .|
|       ..o.*.= ..|
|         .+...o..|
+----[SHA256]-----+
```

Les clés générées seront placées dans le répertoire **~/.ssh/** :

```
trainee@debian11:~$ ls .ssh
id_dsa      id_ecdsa      id_ed25519      id_rsa
```

```
id_dsa.pub  id_ecdsa.pub  id_ed25519.pub  id_rsa.pub
```

Tunnels SSH

Le protocole SSH peut être utilisé pour sécuriser les protocoles tels telnet, pop3 etc.. En effet, on peut créer un *tunnel* SSH dans lequel passe les communications du protocole non-sécurisé.

La commande pour créer un tunnel ssh prend la forme suivante :

```
ssh -N -f compte@hôte -Lport-local:localhost:port_distant
```

Dans votre cas, vous allez créer un tunnel dans votre propre VM entre le port 15023 et le port 23 :

```
trainee@debian11:~$ su -
Password: fenestros
root@debian11:~# ssh -N -f trainee@localhost -L15023:localhost:23
The authenticity of host 'localhost (::1)' can't be established.
ECDSA key fingerprint is SHA256:38O9lqa1AHvviceNEbQ1AOUMspkYBpIpLlu/U05ymqo.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Debian GNU/Linux 11
trainee@localhost's password: trainee
```

Installez maintenant le serveur telnet :

```
root@debian11:~# apt -y install telnetd
```

Vérifiez que le service **inetd** est démarré :

```
root@debian11:~# systemctl status inetd
● inetd.service - Internet superserver
     Loaded: loaded (/lib/systemd/system/inetd.service; enabled; vendor preset: enabl>
```

```
     Active: active (running) since Tue 2022-05-03 11:55:27 CEST; 44s ago
       Docs: man:inetd(8)
   Main PID: 5110 (inetd)
      Tasks: 1 (limit: 4632)
     Memory: 576.0K
        CPU: 7ms
     CGroup: /system.slice/inetd.service
             └─5110 /usr/sbin/inetd

May 03 11:55:27 debian11.ittraining.loc systemd[1]: Starting Internet superserver...
May 03 11:55:27 debian11.ittraining.loc systemd[1]: Started Internet superserver.
```

Connectez-vous ensuite via telnet sur le port 15023, vous constaterez que votre connexion n'aboutit pas :

```
root@debian11:~# telnet localhost 15023
Trying ::1...
Connected to localhost.
Escape character is '^]'.
Debian GNU/Linux 11
debian11.ittraining.loc login: trainee
Password: trainee
Linux debian11.ittraining.loc 5.17.0-1-amd64 #1 SMP PREEMPT Debian 5.17.3-1 (2022-04-18) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue May  3 08:57:59 CEST 2022 from 10.0.2.1 on pts/0

trainee@debian11:~$ whoami
trainee
```

```
trainee@debian11:~$ pwd
/home/trainee

trainee@debian11:~$ exit
logout
Connection closed by foreign host.
```

Important - Notez bien que votre communication telnet passe par le tunnel SSH.

1.5 - SCP

Présentation

La commande **scp** est le successeur et la remplaçante de la commande **rcp** de la famille des commandes **remote**. Il permet de faire des transferts sécurisés à partir d'une machine distante :

```
$ scp compte@numero_ip(nom_de_machine):/chemin_distant/fichier_distant /chemin_local/fichier_local
```

ou vers une machine distante :

```
$ scp /chemin_local/fichier_local compte@numero_ip(nom_de_machine):/chemin_distant/fichier_distant
```

Utilisation

Nous allons maintenant utiliser **scp** pour chercher un fichier sur le «serveur» :

Créez le fichier **/home/trainee/scp_test** :

```
trainee@debian11:~$ touch scp-test
trainee@debian11:~$ exit
logout
Connection closed by foreign host.
root@debian11:~#
```

Récupérez le fichier **scp_test** en utilisant scp :

```
root@debian11:~# scp trainee@127.0.0.1:/home/trainee/scp-test .
The authenticity of host '127.0.0.1 (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:Q7T/CP0SLiMbMAIgVzTuEHegYS/spPE5zzQchCHD5Vw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '127.0.0.1' (ECDSA) to the list of known hosts.
\S
Kernel \r on an \m
trainee@127.0.0.1's password: trainee
scp-test
100%    0     0.0KB/s   00:00

root@debian11:~# ls -l
...
-rw-r--r--. 1 root root    0 Aug 30 03:55 scp-test
...
```

1.6 - Mise en Place des Clefs Asymétriques

Il convient maintenant de se connecter sur le «serveur» en utilisant ssh et vérifiez la présence du répertoire ~/.ssh :

```
root@debian11:~# ssh -l trainee 127.0.0.1
\S
Kernel \r on an \m
trainee@127.0.0.1's password: trainee
```

```
Activate the web console with: systemctl enable --now cockpit.socket

trainee@debian11:~$ ls -la | grep .ssh
drwx------.  2 trainee trainee      4096 Aug 30 02:26 .ssh
```

Important - Si le dossier distant .ssh n'existe pas dans le répertoire personnel de l'utilisateur connecté, il faut le créer avec des permissions de 700. Dans votre cas, puisque votre machine joue le rôle de serveur **et** de client, le dossier /home/trainee/.ssh existe **déjà**.

Ensuite, il convient de transférer le fichier local **.ssh/id_ecdsa.pub** du «client» vers le «serveur» en le renommant en **authorized_keys** :

```
trainee@debian11:~$ exit
logout
Connection to 127.0.0.1 closed.

root@debian11:~# exit
logout

trainee@debian11:~$ scp .ssh/id_ecdsa.pub trainee@127.0.0.1:/home/trainee/.ssh/authorized_keys
The authenticity of host '127.0.0.1 (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:Q7T/CP0SLiMbMAIgVzTuEHegYS/spPE5zzQchCHD5Vw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '127.0.0.1' (ECDSA) to the list of known hosts.
\S
Kernel \r on an \m
trainee@127.0.0.1's password: trainee
id_ecdsa.pub
100%  192   497.6KB/s   00:00
```

Connectez-vous via ssh :

```
trainee@debian11:~$ ssh -l trainee localhost
The authenticity of host 'localhost (::1)' can't be established.
ECDSA key fingerprint is SHA256:Q7T/CP0SLiMbMAIgVzTuEHegYS/spPE5zzQchCHD5Vw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
\S
Kernel \r on an \m
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Mon Aug 30 03:57:14 2021 from 127.0.0.1
trainee@debian11:~$
```

Important - Lors de la connexion au serveur, l'authentification utilise le couple de clefs asymétrique au format ecdsa et aucun mot de passe n'est requis.

Insérez maintenant les clefs publiques restantes dans le fichier .ssh/authorized_keys :

```
trainee@debian11:~$ cd .ssh
[trainee@centos8 .ssh]$ ls
authorized_keys  id_dsa  id_dsa.pub  id_ecdsa  id_ecdsa.pub  id_ed25519  id_ed25519.pub  id_rsa  id_rsa.pub
known_hosts
[trainee@centos8 .ssh]$ cat authorized_keys
ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBHDrzSXP+Ecxf/sQ18VwCRNm7rrSrrsaJmuIw/RgTH5puKF5E+Yy15cvAAKBX
pJPxUmrOaOyhab84PevV7XSHcI= trainee@centos8.ittraining.loc

[trainee@centos8 .ssh]$ cat id_rsa.pub >> authorized_keys
[trainee@centos8 .ssh]$ cat id_dsa.pub >> authorized_keys
[trainee@centos8 .ssh]$ cat id_ed25519.pub >> authorized_keys

[trainee@centos8 .ssh]$ cat authorized_keys
```

```
ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBHDrzSXP+Ecxf/sQ18VwCRNm7rrSrrsaJmuIw/RgTH5puKF5E+Yy15cvAAKBX
pJPxUmrOaOyhab84PevV7XSHcI= trainee@centos8.ittraining.loc
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQD3ZSMn/GIAHtaDFc6ZNnKJam9hzq8TxqMN5IopUr8QhwODyPadbB+FgH4r50qTux4ubwr1BlymgIdqRVWy3
2mE15M8tdtKc3j8DNMpUwPGEh+s/PT7GW+3E3shoyPvpLc1kKaKXKGl/JwfCK/8IYsubk2BmiiJYkzLECotPlaaxm4w1K0AtlnZQuLHt1HK3/rHCh
xo2o2w1t59/QwNcMLiKve1Z+zQ1POKo8VJ/DDrf90y2QWC28ejUs/ZjP6f6C4bn5Jjol4TbHls3ArMsbU6C1Ev5jqbzZ0kmognQ2CnRjeNM51YdFo
6nRsLoPQKpeLRMBpT/87HK1bPUOBVCyhXxSkqkVhMlgg8tgcD/MRlBaUVFoZ1wQ0L26Fe+q7c2Oykj54pdXCAK2ZTpCXGfhd/FxrfqGw7cSeKlGX4
QUzHMjMk2oHFC9h30BUk1gGN21KJPTh7/S1OZVrnCc3GUi5fXPvEpral0IU0sws+j03dj0sWm5ICQFKRkmZN11HCyT0=
trainee@centos8.ittraining.loc
ssh-dss
AAAAB3NzaC1kc3MAAACBALIdwEEqHrMWSUdzARm9ldsZK9ebbtZShtmwgdjphOk77fxymK0y6wV7QEmLL25LOcLb12uZ1F0LtRt/t2oqgrwqk3vUS
pCPLr09AXpcD/nxL9kc+rUxHyl6u1mHtyfCVLCPSvavCMR8TaA8egVMk3EwGRfHTiuDOKi7Iwus7gXPAAAAFQDHEQPGVRI7gVYKzCT6nrjDsQQ6jw
AAAIEAhhhH7fEjdldASXY0qTWkCvcs3cfK9/Ff315zByn47O02y9Vdo3QG5nOr1Oo8fc2xEkIBNmFr8Rr2g60cpvEev5hy4XZ1ghxnQ53iwKuiS72
ZATwhD6bZBrsiH0k1Et25gRcj5KCvDe/jHhbxCxsCuHUH2qvWsQNVwztE7hD0sxkAAACAQ8Dkpy8zXj7jW8o1txxf2W6J4r2+1lPldymA45ywZokN
4SCwvXlpPAuyBt0/HiU0R2PI9aqOAMosCLcy9WmnSwLQ2Z7QcD2i3XlAih2+1q9NJP22sPT3jSK9UZcdRjoZ/eNiz84sXZucNape32tFxjvcV4txo
bH/vD53q8g63fA= trainee@centos8.ittraining.loc
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOfFQULLU8IZyKiSU63D2Zz6yGLqyHcBHnCRdSR9JSmc trainee@centos8.ittraining.loc
```

1.7 - Services réseaux

Quand un client émet une demande de connexion vers une application réseau sur un serveur, il utilise un socket attaché à un port local **supérieur à 1023**, alloué d'une manière dynamique. La requête contient le port de destination sur le serveur. Certaines applications serveurs se gèrent toutes seules, ce qui est la cas par exemple d'**apache2**. Par contre d'autres sont gérées par le service **xinetd**.

inetd

Le programme inetd est configuré via le fichier **/etc/inetd.conf** :

```
root@debian11:~# cat /etc/inetd.conf
# /etc/inetd.conf:  see inetd(8) for further informations.
```

```
#
# Internet superserver configuration database
#
#
# Lines starting with "#:LABEL:" or "#<off>#" should not
# be changed unless you know what you are doing!
#
# If you want to disable an entry so it isn't touched during
# package updates just comment it out with a single '#' character.
#
# Packages should modify this file by using update-inetd(8)
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
#:INTERNAL: Internal services
#discard                stream  tcp     nowait  root    internal
#discard                dgram   udp     wait    root    internal
#daytime                stream  tcp     nowait  root    internal
#time           stream  tcp     nowait  root    internal

#:STANDARD: These are standard services.
telnet          stream  tcp     nowait  telnetd /usr/sbin/tcpd  /usr/sbin/in.telnetd

#:BSD: Shell, login, exec and talk are BSD protocols.

#:MAIL: Mail, news and uucp services.

#:INFO: Info services

#:BOOT: TFTP service is provided primarily for booting.  Most sites
#       run this only on machines acting as "boot servers."

#:RPC: RPC based services
```

```
#:HAM-RADIO: amateur-radio services

#:OTHER: Other services
```

Les lignes de configuration des serveurs ressemblent à :

```
telnet          stream  tcp     nowait  telnetd /usr/sbin/tcpd  /usr/sbin/in.telnetd
```

Le premier champs de la ligne identifie le nom du port qui identifie l'application. Inetd lit ensuite le fichier **/etc/services** pour déduire le numéro de port concerné et se met à l'écoute de celui-ci.

Le deuxième et le troisième champs définissent le type de protocole, à savoir:

- stream tcp pour le tcp
- dgram udp pour l'udp

Le quatrième champs prend un de deux valeurs:

- nowait
  - indique qu'il y aura un serveur par client
- wait
  - indique qu'il y aura un seul serveur pour l'ensemble des clients

Le cinquième champs indique l'identité sous laquelle sera exécuté le serveur.

Le sixième champs indique l'exécutable. Dans ce cas c'est **/usr/sbin/tcpd**.

Le septième champs indique les arguments de l'application serveur dont l'argument **0** est le nom de l'application.

TCP Wrapper

Lors de l'utilisation d'inetd, **TCP Wrapper** est utilisé pour contrôler l'accès à des services réseaux grâce à des **ACL** :

```
telnet          stream  tcp     nowait  telnetd /usr/sbin/tcpd  /usr/sbin/in.telnetd
```

Quand une requête arrive pour le serveur telnet, inetd active le wrapper **tcpd** au lieu d'activer **/usr/sbin/in.telnetd** directement.

tcpd met à jour un journal et vérifie si le client a le droit d'utiliser le service concerné. Les ACL se trouvent dans deux fichiers:

```
root@debian11:~# cat /etc/hosts.allow
# /etc/hosts.allow: list of hosts that are allowed to access the system.
#                   See the manual pages hosts_access(5) and hosts_options(5).
#
# Example:    ALL: LOCAL @some_netgroup
#             ALL: .foobar.edu EXCEPT terminalserver.foobar.edu
#
# If you're going to protect the portmapper use the name "rpcbind" for the
# daemon name. See rpcbind(8) and rpc.mountd(8) for further information.
#

root@debian11:~# cat /etc/hosts.deny
# /etc/hosts.deny: list of hosts that are _not_ allowed to access the system.
#                  See the manual pages hosts_access(5) and hosts_options(5).
#
# Example:    ALL: some.host.name, .some.domain
#             ALL EXCEPT in.fingerd: other.host.name, .other.domain
#
# If you're going to protect the portmapper use the name "rpcbind" for the
# daemon name. See rpcbind(8) and rpc.mountd(8) for further information.
#
# The PARANOID wildcard matches any host whose name does not match its
# address.
#
# You may wish to enable this to ensure any programs that don't
# validate looked up hostnames still leave understandable logs. In past
# versions of Debian this has been the default.
```

```
# ALL: PARANOID
```

Il faut noter que si ces fichiers n'existent pas ou sont vides, il n'y a pas de contrôle d'accès.

Le format d'une ligne dans un de ces deux fichiers est:

```
démon: liste_de_clients
```

Par exemple dans le cas de notre serveur telnetd, une ligne dans le fichier **/etc/hosts.allow** similaire à:

```
in.telnetd: 192.168.1.10, .fenestros.com
```

implique que la machine dont le numéro IP est le 192.168.1.10 ainsi que les machines du domaine **fenestros.com** sont autorisées à utiliser le service.

Le mot clef **ALL** peut être utilisé pour indiquer tout. Par exemple, **ALL:ALL** dans le fichier **/etc/host.deny** bloque effectivement toute tentative de connexion à un service inetd sauf pour les ACL inclus dans le fichier /etc/host.allow.
