Version : **2026.01**

Dernière mise-à-jour : 2025/12/11 15:19

LDF408 - Cryptologie

Contenu du module

Le Problématique

Le **sniffing** des paquets de données est possible sur un réseau utilisant une technologie de diffusion tel un réseau **Ethernet**. En effet certains protocoles ne cryptent pas les données avant de les envoyer dont :

- Telnet,
- Rlogin,
- Ftp,
- Pop3.

Un *sniffeur* est un logiciel qui captent les paquets circulant sur un réseau de type diffusion afin de les analyser. Le logiciel les plus utilisé est :

- Tcpdump.

LAB #1 - Utilisation de tcpdump

Le logiciel **tcpdump** sert à écouter le réseau en interceptant les paquets.

1.1 - Utilisation

Installez **tcpdump** :

```
root@debian12:~# apt install tcpdump
```

L'option -i

Pour écouter sur une **interface spécifique**, utilisez l'option **-i** :

```
root@debian12:~# tcpdump -i ens18 -c 10
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens18, link-type EN10MB (Ethernet), snapshot length 262144 bytes
17:29:37.411395 IP 10.0.2.46.ssh > 10.0.2.1.42252: Flags [P.], seq 119569853:119570041, ack 3811519177, win 501,
```

```
options [nop,nop,TS val 3553055771 ecr 1647876767], length 188
17:29:37.411528 IP 10.0.2.1.42252 > 10.0.2.46.ssh: Flags [.], ack 188, win 10591, options [nop,nop,TS val
1647876791 ecr 3553055771], length 0
17:29:37.493738 IP 10.0.2.46.36881 > dns.google.domain: 55995+ PTR? 1.2.0.10.in-addr.arpa. (39)
17:29:37.495598 IP dns.google.domain > 10.0.2.46.36881: 55995 NXDomain 0/0/0 (39)
17:29:37.495837 IP 10.0.2.46.33051 > dns.google.domain: 43187+ PTR? 46.2.0.10.in-addr.arpa. (40)
17:29:37.497464 IP dns.google.domain > 10.0.2.46.33051: 43187 NXDomain 0/0/0 (40)
17:29:37.497758 IP 10.0.2.46.ssh > 10.0.2.1.42252: Flags [P.], seq 188:536, ack 1, win 501, options [nop,nop,TS
val 3553055857 ecr 1647876791], length 348
17:29:37.497891 IP 10.0.2.1.42252 > 10.0.2.46.ssh: Flags [.], ack 536, win 10591, options [nop,nop,TS val
1647876877 ecr 3553055857], length 0
17:29:37.597097 IP 10.0.2.46.52033 > dns.google.domain: 12468+ PTR? 8.8.8.8.in-addr.arpa. (38)
17:29:37.598847 IP dns.google.domain > 10.0.2.46.52033: 12468 1/0/0 PTR dns.google. (62)
10 packets captured
14 packets received by filter
0 packets dropped by kernel
```

Notez qu'à la fin, un résumé vous est présenté, par exemple :

```
...
10 packets captured
14 packets received by filter
0 packets dropped by kernel
```

Important : L'option **-c** limite le nombre de paquets capturés.

L'option -x

Pour écouter sur une interface spécifique et voir le contenu en Hexadécimal, utilisez les options -i et **-x** :

```
root@debian12:~# tcpdump -i ens18 -x -c 3
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens18, link-type EN10MB (Ethernet), snapshot length 262144 bytes
17:27:24.043320 IP 10.0.2.46.ssh > 10.0.2.1.42252: Flags [P.], seq 119567701:119567889, ack 3811519061, win 501,
options [nop,nop,TS val 3552922403 ecr 1647743397], length 188
        0x0000:  4510 00f0 84b8 4000 4006 9d11 0a00 022e
        0x0010:  0a00 0201 0016 a50c 0720 7555 e32f 2a55
        0x0020:  8018 01f5 1911 0000 0101 080a d3c5 4b23
        0x0030:  6236 91a5 0000 00b0 77fb cb1f c046 bb66
        0x0040:  c32f 923b a994 d49b f063 5539 130e 764b
        0x0050:  06fe b5be f2d1 7cee ef79 8d3e ec6e 1e7f
        0x0060:  c296 1fbf f4e8 67b0 f16d 5d98 1963 424c
        0x0070:  f6de 9287 5dbe 98c0 2b95 05cb 37f3 a653
        0x0080:  dbff 81a1 d03f c288 bbaf 4756 41be 64ea
        0x0090:  706a 55ff 8322 e32e ea56 2e60 5210 43e6
        0x00a0:  9eec 5bb1 e519 1936 4ee7 809b 6a18 675e
        0x00b0:  29b7 de38 921d a543 4ca7 7132 ff0f e399
        0x00c0:  5338 304d 03a3 beed ddbe 4530 cb71 fea2
        0x00d0:  618b b56d b45c 9ab6 9b71 563e 058d de9a
        0x00e0:  d249 d57e 7a53 f6b5 7bf6 5924 46f0 6c74
17:27:24.043485 IP 10.0.2.1.42252 > 10.0.2.46.ssh: Flags [.], ack 188, win 10548, options [nop,nop,TS val
1647743423 ecr 3552922403], length 0
        0x0000:  4510 0034 41fb 4000 4006 e08a 0a00 0201
        0x0010:  0a00 022e a50c 0016 e32f 2a55 0720 7611
        0x0020:  8010 2934 1855 0000 0101 080a 6236 91bf
        0x0030:  d3c5 4b23
17:27:24.125464 IP 10.0.2.46.42246 > dns.google.domain: 35092+ PTR? 1.2.0.10.in-addr.arpa. (39)
        0x0000:  4500 0043 3869 4000 4011 e603 0a00 022e
        0x0010:  0808 0808 a506 0035 002f 1c7e 8914 0100
        0x0020:  0001 0000 0000 0000 0131 0132 0130 0231
        0x0030:  3007 696e 2d61 6464 7204 6172 7061 0000
        0x0040:  0c00 01
3 packets captured
10 packets received by filter
```

```
0 packets dropped by kernel
```

L'option -X

Pour écouter sur une interface spécifique et voir le contenu en Hexadécimal et en ASCII, utilisez les options -i et **-X** :

```
root@debian12:~# tcpdump -i ens18 -X -c 3
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens18, link-type EN10MB (Ethernet), snapshot length 262144 bytes
13:07:32.922306 IP 10.0.2.46.ssh > 10.0.2.1.42252: Flags [P.], seq 119572449:119572637, ack 3811519901, win 501,
options [nop,nop,TS val 3553115863 ecr 1647936856], length 188
        0x0000:  4510 00f0 84d9 4000 4006 9cf0 0a00 022e  E.....@.@.......
        0x0010:  0a00 0201 0016 a50c 0720 87e1 e32f 2d9d  ............/-.
        0x0020:  8018 01f5 1911 0000 0101 080a d3c8 3ed7  ..............>.
        0x0030:  6239 8558 0000 00b0 daef 47be bc64 8630  b9.X......G..d.0
        0x0040:  968c 107d ba16 f8fb 45db 05ff e566 a1af  ...}....E....f..
        0x0050:  728f e777 583b 64ac 09a6 f099 c570 ad4f  r..wX;d......p.O
        0x0060:  17d1 afa8 dbb3 1d59 122a 3e9d e4be 07f7  .......Y.*>.....
        0x0070:  7b85 75fc 49f4 e0fe 37ab f924 acc6 4f43  {.u.I...7..$..OC
        0x0080:  6231 f469 e3b2 ebb9 e9a4 6bfb cd89 66e2  b1.i......k...f.
        0x0090:  9fda ccf9 39ad 272a f373 167e 13e2 b56c  ....9.'*.s.~...l
        0x00a0:  3625 f2e2 7898 d061 6191 d5c4 a268 b1e0  6%..x..aa....h..
        0x00b0:  1f4a a1f9 e319 18c9 e55a b700 e281 1f71  .J.......Z.....q
        0x00c0:  06fb 4e11 1145 23e2 a194 d91c 8e22 f8ef  ..N..E#......"..
        0x00d0:  5e96 6e34 f24f 1b67 754b 4c7e 5e51 a2cf  ^.n4.O.guKL~^Q..
        0x00e0:  e2ed c5bb 409e eae3 c905 54ab cfc2 5a55  ....@.....T...ZU
13:07:32.922462 IP 10.0.2.1.42252 > 10.0.2.46.ssh: Flags [.], ack 188, win 10611, options [nop,nop,TS val
1647936883 ecr 3553115863], length 0
        0x0000:  4510 0034 4232 4000 4006 e053 0a00 0201  E..4B2@.@..S....
        0x0010:  0a00 022e a50c 0016 e32f 2d9d 0720 889d  ........./-.....
        0x0020:  8010 2973 1855 0000 0101 080a 6239 8573  ..)s.U......b9.s
        0x0030:  d3c8 3ed7                                ..>.
13:07:33.008324 IP 10.0.2.46.47533 > dns.google.domain: 49115+ PTR? 1.2.0.10.in-addr.arpa. (39)
```

```
        0x0000:  4500 0043 2e6b 4000 4011 f001 0a00 022e  E..C.k@.@.......
        0x0010:  0808 0808 b9ad 0035 002f 1c7e bfdb 0100  .......5./.~....
        0x0020:  0001 0000 0000 0000 0131 0132 0130 0231  .........1.2.0.1
        0x0030:  3007 696e 2d61 6464 7204 6172 7061 0000  0.in-addr.arpa..
        0x0040:  0c00 01                                  ...
3 packets captured
10 packets received by filter
0 packets dropped by kernel
```

L'option -w

Pour écouter sur une interface spécifique et envoyer la sortie dans un fichier, utilisez les options -i et **-w** et patientez 5 minutes :

```
root@debian12:~# tcpdump -i ens18 -w log.dump
tcpdump: listening on ens18, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C42 packets captured
45 packets received by filter
0 packets dropped by kernel

root@debian12:~# ls -l log.dump
-rw-r--r-- 1 tcpdump tcpdump 25555 Nov 28 13:11 log.dump
```

Important - Arrêtez la sortie de la commande à l'aide des touches **^C**.

Notez que le fichier log.dump est au format **libpcap** et non au format texte. Il est donc inutile d'essayer de lire son contenu avec une commande telle **cat** :

```
root@debian12:~# file log.dump
log.dump: pcap capture file, microsecond ts (little-endian) - version 2.4 (Ethernet, capture length 262144)
```

L'option -v

Tcpdump peut être utilisé avec un de trois modes verbose.

Mode	Option
Light verbose	-v
Medium verbose	-vv
Full verbose	-vvv

```
root@debian12:~# tcpdump -i ens18 -v -c 3
tcpdump: listening on ens18, link-type EN10MB (Ethernet), snapshot length 262144 bytes
13:13:22.869956 IP (tos 0x10, ttl 64, id 34138, offset 0, flags [DF], proto TCP (6), length 176)
    10.0.2.46.ssh > 10.0.2.1.42252: Flags [P.], cksum 0x18d1 (incorrect -> 0x3397), seq 119580817:119580941, ack
3811523793, win 501, options [nop,nop,TS val 3553465811 ecr 1648286807], length 124
13:13:22.870085 IP (tos 0x10, ttl 64, id 17171, offset 0, flags [DF], proto TCP (6), length 52)
    10.0.2.1.42252 > 10.0.2.46.ssh: Flags [.], cksum 0x1855 (incorrect -> 0x174a), ack 124, win 10660, options
[nop,nop,TS val 1648286831 ecr 3553465811], length 0
13:13:22.951837 IP (tos 0x10, ttl 64, id 34139, offset 0, flags [DF], proto TCP (6), length 176)
    10.0.2.46.ssh > 10.0.2.1.42252: Flags [P.], cksum 0x18d1 (incorrect -> 0x5f7e), seq 124:248, ack 1, win 501,
options [nop,nop,TS val 3553465893 ecr 1648286831], length 124
3 packets captured
10 packets received by filter
0 packets dropped by kernel
```

1.2 - Filtrage à l'écoute

Tcpdump peut effectuer du filtrage lors de l'écoute.

Pour uniquement écouter les paquets en provenance de l'adresse IP 192.168.1.11, utilisez la condition **src host** :

```
# tcpdump src host 192.168.1.11 [Entrée]
```

Pour uniquement écouter les paquets en provenance de l'adresse IP 192.168.1.11 et vers l'adresse 192.168.1.2, utilisez les conditions src host et **dst host** :

```
# tcpdump src host 192.168.1.11 and dst host 192.168.1.2 [Entrée]
```

Pour uniquement écouter les paquets d'un port précis, utilisez la condition **port** :

```
# tcpdump -i eth0 port 80 [Entrée]
```

Pour uniquement écouter les paquets d'un protocole précis, utilisez une condition telle **ip**, **icmp**, **arp**, **rarp**, **udp** ou **tcp**:

```
# tcpdump -i eth0 udp [Entrée]
```

Pour uniquement écouter les paquets d'une taille inférieure à 100 octets, utilisez la condition **less** :

```
# tcpdump -i eth0 less 100 [Entrée]
```

Pour uniquement écouter les paquets d'une taille supérieure à 100 octets, utilisez la condition **great** :

```
# tcpdump -i eth0 greater 100 [Entrée]
```

L'utilisation des ses options et conditions peut être combinée pour donner des commandes telles :

```
# tcpdump -i eth0 -X src host 192.168.1.11 and dst host 192.168.1.2 and port 21 and ftp [Entrée]
```

Options de la commande

Les options de cette commande sont :

```
root@debian12:~# tcpdump --help
tcpdump version 4.99.3
libpcap version 1.10.3 (with TPACKET_V3)
```

```
OpenSSL 3.0.17 1 Jul 2025
Usage: tcpdump [-AbdDefhHIJKlLnNOpqStuUvxX#] [ -B size ] [ -c count ] [--count]
                [ -C file_size ] [ -E algo:secret ] [ -F file ] [ -G seconds ]
                [ -i interface ] [ --immediate-mode ] [ -j tstamptype ]
                [ -M secret ] [ --number ] [ --print ] [ -Q in|out|inout ]
                [ -r file ] [ -s snaplen ] [ -T type ] [ --version ]
                [ -V file ] [ -w file ] [ -W filecount ] [ -y datalinktype ]
                [ --time-stamp-precision precision ] [ --micro ] [ --nano ]
                [ -z postrotate-command ] [ -Z user ] [ expression ]
```

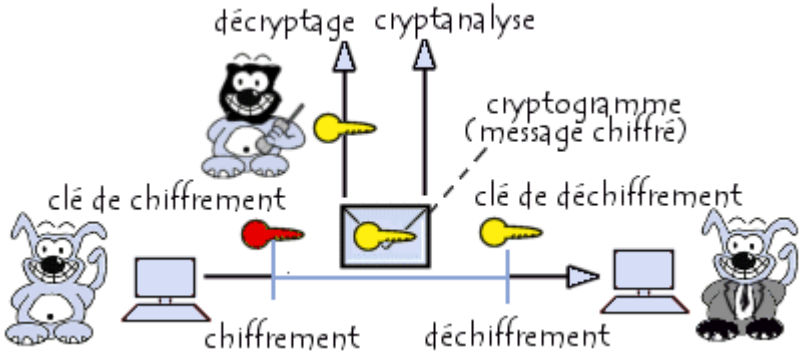
Les Contre-Mesures

Les contre-mesures incluent l'utilisation du chiffrement sous la forme de SSH, de SCP et d'OpenVPN.

Introduction à la cryptologie

Définitions

- **La Cryptologie**
  - La science qui étudie les aspects scientifiques de ces techniques, c'est-à-dire qu'elle englobe la cryptographie et la cryptanalyse.
- **La Cryptanalyse**
  - Lorsque la clef de déchiffrement n'est pas connue de l'attaquant on parle alors de cryptanalyse ou cryptoanalyse (on entend souvent aussi le terme plus familier de cassage).
- **La Cryptographie**
  - Un terme générique désignant l'ensemble des techniques permettant de chiffrer des messages, c'est-à-dire permettant de les rendre inintelligibles sans une action spécifique. Les verbes crypter et chiffrer sont utilisés.
- **Le Décryptement ou Décryptage**
  - Est le fait d'essayer de déchiffrer illégitimement le message (que la clé de déchiffrement soit connue ou non de l'attaquant).



La Cryptographie

La cryptographie apporte quatre points clefs:

- La confidentialité
  - consiste à rendre l'information inintelligible à d'autres personnes que les acteurs de la transaction.
- L'intégrité
  - consiste à déterminer si les données n'ont pas été altérées durant la communication (de manière fortuite ou intentionnelle).
- L'authentification
  - consiste à assurer l'identité d'un utilisateur.
- La non-répudiation
  - est la garantie qu'aucun des correspondants ne pourra nier la transaction.

La cryptographie est basée sur l'arithmétique. Il s'agit, dans le cas d'un texte, de transformer les lettres qui composent le message en une succession de chiffres (sous forme de bits dans le cas de l'informatique), puis ensuite de faire des calculs sur ces chiffres pour:

- Procéder au chiffrement
  - Le résultat de cette modification (le message chiffré) est appelé cryptogramme (Ciphertext) par opposition au message initial, appelé message en clair (Plaintext)
- Procéder au déchiffrement

Le chiffrement se fait à l'aide d'une clef de chiffrement. Le déchiffrement nécessite une clef de déchiffrement.

On distingue deux types de clefs:

- Les clés symétriques:
  - des clés utilisées pour le chiffrement ainsi que pour le déchiffrement. On parle alors de chiffrement symétrique ou de chiffrement à clé secrète.
- Les clés asymétriques:
  - des clés utilisées dans le cas du chiffrement asymétrique (aussi appelé chiffrement à clé publique). Dans ce cas, une clé différente est utilisée pour le chiffrement et pour le déchiffrement.

Le Chiffrement par Substitution

Le chiffrement par substitution consiste à remplacer dans un message une ou plusieurs entités (généralement des lettres) par une ou plusieurs autres entités. On distingue généralement plusieurs types de cryptosystèmes par substitution :

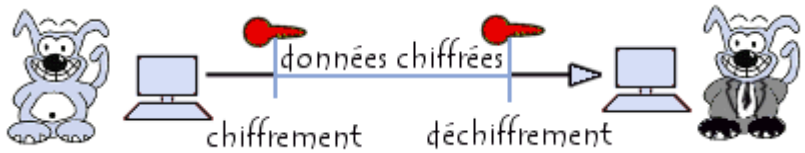
- La substitution **monoalphabétique**
  - consiste à remplacer chaque lettre du message par une autre lettre de l'alphabet
- La substitution **polyalphabétique**
  - consiste à utiliser une suite de chiffres monoalphabétique réutilisée périodiquement
- La substitution **homophonique**
  - permet de faire correspondre à chaque lettre du message en clair un ensemble possible d'autres caractères
- La substitution de **polygrammes**
  - consiste à substituer un groupe de caractères (polygramme) dans le message par un autre groupe de caractères

Algorithmes à clé secrète

Le Chiffrement Symétrique

Ce système est aussi appelé le système à **Clef Secrète** ou à **clef privée**.

Ce système consiste à effectuer une opération de chiffrement par algorithme mais comporte un inconvénient, à savoir qu'il nécessite un canal sécurisé pour la transmission de la clef de chiffrement/déchiffrement.



Important - Le système de Méthode du Masque Jetable (One Time Pad) fût mis au point dans les années 1920. Il utilisait une clef générée aléatoirement à usage unique.

Les algorithmes de chiffrement symétrique couramment utilisés en informatique sont:

- **Data Encryption Standard** (DES),
- **Triple DES** (3DES),
- **RC2**,
- **Blowfish**,
- **International Data Encryption Algorithm** (IDEA),
- **Advanced Encryption Standard** (AES).

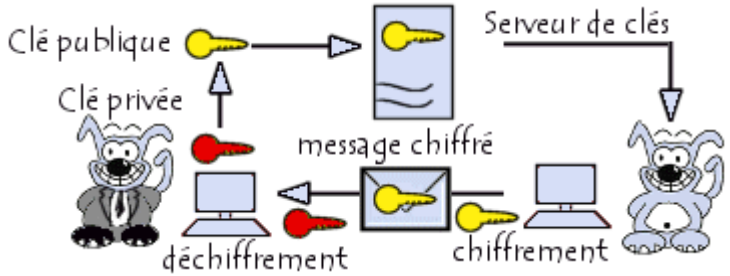
Algorithmes à clef publique

Le Chiffrement Asymétrique

Ce système est aussi appelé **Système à Clef Publique**.

Ce système consiste à avoir deux clefs appelées des **bi-clefs**:

- Une clef **publique** pour le chiffrement
- Une clef **secrète** ou **privée** pour le déchiffrement



- L'utilisateur A (celui qui déchiffre) choisit une clef privée.
- A partir de cette clef il génère plusieurs clefs publiques grâce à un algorithme.
- L'utilisateur B (celui qui chiffre) choisit une des clefs publiques à travers un canal non-sécurisé pour chiffrer les données à l'attention de l'utilisateur A.

Ce système est basé sur ce que l'on appelle une **fonction à trappe à sens unique** ou **one-way trap door**.

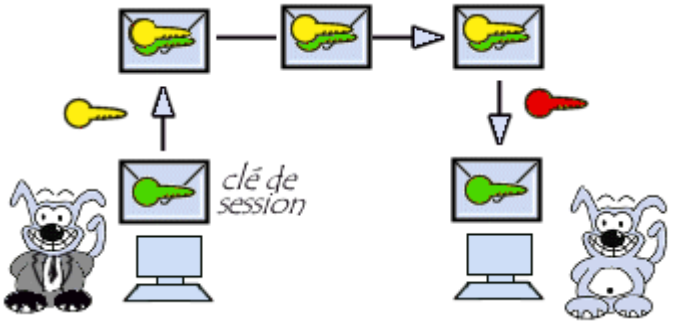
Il existe toutefois un problème – s'assurer que la clef publique récupérée est bien celle qui correspond au destinataire !

Les algorithmes de chiffrement asymétrique couramment utilisés en informatique sont:

- **Digital Signature Algorithm** (DSA)
- **Rivest, Shamir, Adleman** (RSA)

La Clef de Session

Ce système est un compromis entre le système symétrique et le système asymétrique. Il permet l'envoie de données chiffrées à l'aide d'un algorithme de chiffrement symétrique par un canal non-sécurisé et a été mis au point pour palier au problème de lenteur de déchiffrement du système asymétrique.

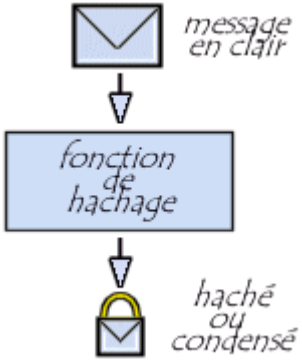


Ce système fonctionne de la façon suivante :

- L'utilisateur A chiffre une clef privée générée aléatoirement, appelée une « clef de session », en utilisant une des clefs publiques de l'utilisateur B.
- L'utilisateur A chiffre les données avec la clef de session.
- L'utilisateur B déchiffre la clef de session en utilisant sa propre clef privée.
- L'utilisateur B déchiffre les données en utilisant la clef de session.

Fonctions de Hachage

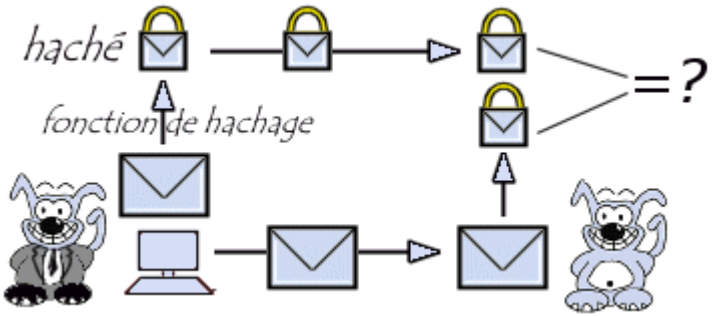
La fonction de **hachage**, aussi appelée une fonction de **condensation**, est à **sens unique** (one way function). Il « condense » un message en clair et produit un haché unique.



Les deux algorithmes de hachage utilisés sont:

- **Message Digest 5** (MD5)
- **Secure Hash Algorithm** (SHA)

Lors de son envoie, le message est accompagné de son haché et il est donc possible de garantir son intégrité:

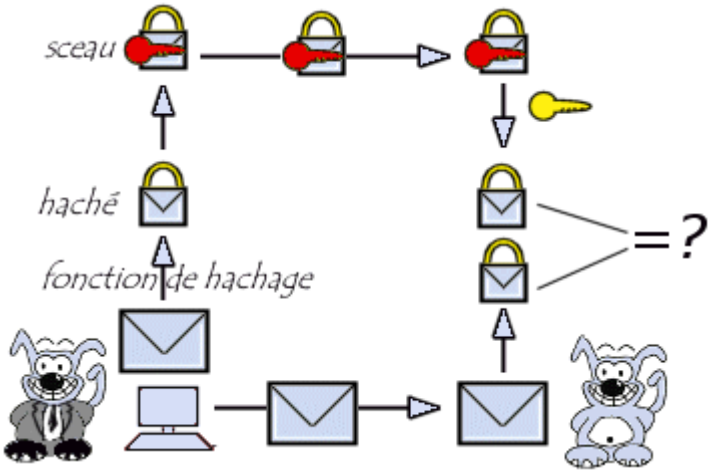


- A la réception du message, le destinataire ou l'utilisateur B calcule le haché du message reçu et le compare avec le haché accompagnant le document.
- Si le message ou le haché a été falsifié durant la communication, les deux empreintes ne correspondront pas.

Important - Ce système permet de vérifier que l'empreinte correspond bien au message reçu, mais ne permet pas de prouver que le message a bien été envoyé par l'utilisateur A.

Signature Numérique

Pour garantir l'authentification du message l'utilisateur A va chiffrer ou **signer** le haché à l'aide de sa clé privée. Le haché signé est appelé un **sceau**.



- L'utilisateur A envoie le sceau au destinataire.
- A la réception du message L'utilisateur B déchiffre le sceau avec la clé publique de l'utilisateur A.
- Il compare le haché obtenu au haché reçu en pièce jointe.

Ce mécanisme de création de sceau est appelé **scellement**.

Ce mécanisme est identique au procédé utilisé par SSH lors d'une connexion

PKI

On appelle **PKI** (Public Key Infrastucture, ou en français **infrastructure à clé publique (ICP)**, parfois **infrastructure de gestion de clés (IGC)**) l'ensemble des solutions techniques basées sur la cryptographie à clé publique.

Les cryptosystèmes à clés publiques permettent de s'affranchir de la nécessité d'avoir recours systématiquement à un canal sécurisé pour s'échanger les clés. En revanche, la publication de la clé publique à grande échelle doit se faire en toute confiance pour assurer que :

- La clé publique est bien celle de son propriétaire ;
- Le propriétaire de la clé est digne de confiance ;
- La clé est toujours valide.

Ainsi, il est nécessaire d'associer au bi-clé (ensemble clé publique / clé privée) un certificat délivré par un **tiers de confiance** : l'infrastructure de gestion de clés.

Le tiers de confiance est une entité appelée communément autorité de certification (ou en anglais Certification authority, abrégé CA) chargée d'assurer la véracité des informations contenues dans le certificat de clé publique et de sa validité.

Pour ce faire, l'autorité signe le certificat de clé publique à l'aide de sa propre clé en utilisant le principe de signature numérique.

Le rôle de l'infrastructure de clés publiques est multiple et couvre notamment les champs suivants :

- enregistrer des demandes de clés en vérifiant l'identité des demandeurs ;
- générer les paires de clés (clé privée / clé publique) ;
- garantir la confidentialité des clés privées correspondant aux clés publiques ;
- certifier l'association entre chaque utilisateurs et sa clé publique ;
- révoquer des clés (en cas de perte par son propriétaire, d'expiration de sa date de validité ou de compromission).

Une infrastructure à clé publique est en règle générale composée de trois entités distinctes :

- L'autorité d'enregistrement (AE ou RA pour Recording authority), chargée des formalité administratives telles que la vérification de l'identité des demandeurs, le suivi et la gestion des demandes, etc.) ;
- L'autorité de certification (AC ou CA pour Certification Authority), chargée des tâches techniques de création de certificats. L'autorité de certification est ainsi chargée de la signature des demandes de certificat (CSR pour Certificate Signing Request, parfois appelées PKCS#10, nom du format correspondant). L'autorité de certification a également pour mission la signature des listes de révocations (CRL pour Certificate Revocation List) ;
- L'Autorité de dépôt (Repository) dont la mission est de conserver en sécurité les certificats.

Certificats X509

Pour palier aux problèmes liés à des clefs publiques piratées, un système de certificats a été mis en place.

Le certificat permet d’associer la clef publique à une entité ou une personne. Les certificats sont délivrés par des Organismes de Certification.

Les certificats sont des fichiers divisés en deux parties :

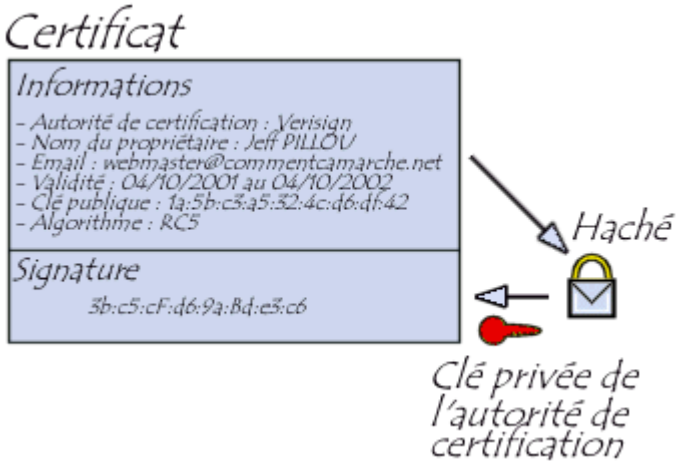
- La partie contenant les informations
- La partie contenant la signature de l'autorité de certification

La structure des certificats est normalisée par le standard **X.509** de l'**Union internationale des télécommunications**.

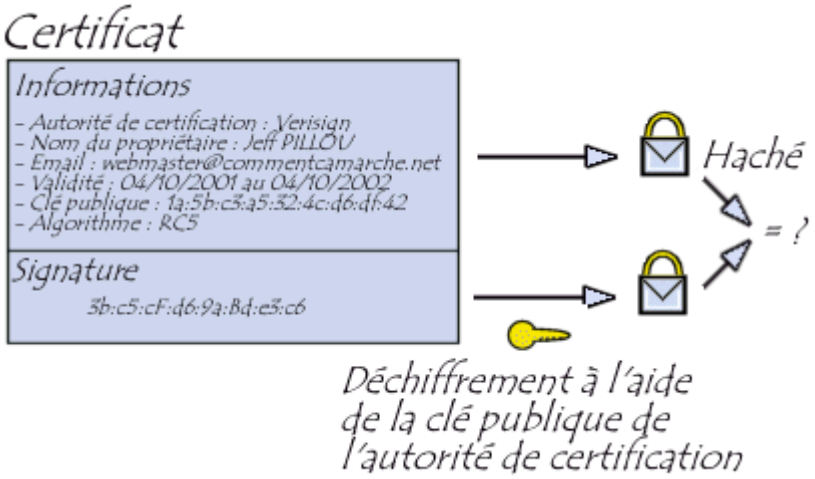
Elle contient :

- Le nom de l'autorité de certification
- Le nom du propriétaire du certificat
- La date de validité du certificat
- L'algorithme de chiffrement utilisé
- La clé publique du propriétaire

Le Certificat est signé par l'autorité de certification:



La vérification se passe ainsi:



LAB #2 - Utilisation de GnuPG

2.1 - Présentation

GNU Privacy Guard permet aux utilisateurs de transférer des messages chiffrés et/ou signés.

2.2 - Utilisation

Pour initialiser GnuPG, saisissez la commande suivante :

```
root@debian12:~# gpg
gpg: directory '/root/.gnupg' created
gpg: keybox '/root/.gnupg/pubring.kbx' created
gpg: WARNING: no command supplied.  Trying to guess what you mean ...
gpg: Go ahead and type your message ...
^C
```

```
gpg: signal Interrupt caught ... exiting
```

Important - Notez l'utilisation des touches **^C** après la ligne **gpg: Go ahead and type your message ...**.

Pour générer les clefs, saisissez la commande suivante :

Important - Lorsque le système vous demande une Passphrase, saisissez une valeur que n'allez **PAS** oublié.

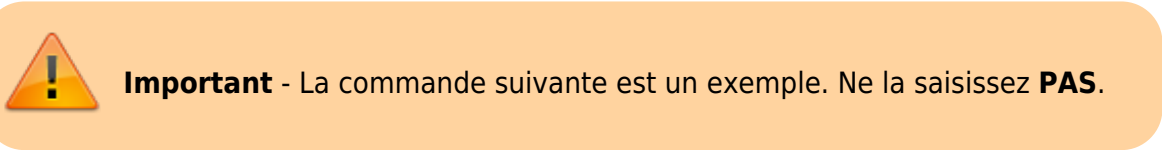
```
root@debian12:~# gpg --full-generate-key
gpg (GnuPG) 2.2.40; Copyright (C) 2022 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
  (14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072)
Requested keysize is 3072 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
```

```
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: ITTRAINING
Email address: infos@ittraining.team
Comment: Test key
You selected this USER-ID:
    "ITTRAINING (Test key) <infos@ittraining.team>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: directory '/root/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/root/.gnupg/openpgp-
revocs.d/B6022CC107539B4036A90FAEABAD13CD27F9E686.rev'
public and secret key created and signed.

pub   rsa3072 2025-11-28 [SC]
      B6022CC107539B4036A90FAEABAD13CD27F9E686
uid                      ITTRAINING (Test key) <infos@ittraining.team>
sub   rsa3072 2025-11-28 [E]
```

La liste de clefs peut être visualisée avec la commande suivante :

```
root@debian12:~# gpg --list-keys
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
/root/.gnupg/pubring.kbx
------------------------
pub   rsa3072 2025-11-28 [SC]
      B6022CC107539B4036A90FAEABAD13CD27F9E686
uid           [ultimate] ITTRAINING (Test key) <infos@ittraining.team>
sub   rsa3072 2025-11-28 [E]
```

Pour importer la clef d'un correspondant dans sa trousse de clefs il convient d'utiliser la commande suivante :




```
# gpg --import la-clef.asc
```

Pour exporter sa clef publique, il convient d'utiliser la commande suivante :

```
root@debian12:~# gpg --export --armor ITTRAINING > ~/mykey.asc

root@debian12:~# cat mykey.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQGNBGkpk+gBDACq6M7rUNQFu/R6J+1p3RAB1+gwnszs/jZuBAo6y9i1buBsySP9
oV9JmFfRe4P2QG/mgmSaGgeO0sE5m+r2Jhif2fShjHYLd6VTSVZRyfO+NW3MDbkm
MtIf4LlXRrTALGE5TclLWFz5a2iqRjtT8IjSPAu1M9TLUaMXtWXN6jQY1YOnjxPN
3HL8bwdaY0k8icKr8JRbmEijCWo2F4t2qTtdOXuRFAImxBpX49eJR0oC5bzXZPYx
LbirNsEwSULIyQ71gdF5OUb0aOSiQLXLuTLAs2BnxAJ82tB/dM8qP0ez0lXLJvvF
```

```
+t0eHdWzUHH4qlXuNtBXK7pEbvjqftO69PJLDp/PwjEfUsELcxDyUFpphhZDJ8zN
qvRgll3fojjr91WWWqr8YTd4EiTvq9KfUyAiTOfKm8k4iRQRTJir267Fq8Mno8/B
My1tNeHWzCXMs3k/GKyi36Uu15wY50hRTPDYexa+aou0QEs4iY3EXjzaOD3BVvRQ
XTBUX/CS3ZmLC3MAEQEAAbQtSVRUUkFJTklORyAoVGVzdCBrZXkpIDxpbmZvc0Bp
dHRyYWluaW5nLnRlYW0+iQHOBBMBCgA4FiEEtgIswQdTm0A2qQ+uq60TzSf55oYF
Amkpk+gCGwMFCwkIBwIGFQoJCAsCBBYCAwECHgECF4AACgkQq60TzSf55oZEsAv+
Ky14vHWjMzU1ieE+XStqGYfjfAHhaNrpkWCz6miozuh+ESifNWpJd96bzwgNLGob
E5mA9oja5jyAEQUGT+gEwgvkbYVe4sf4UcXPRrvDqKiEOkN5rra3kYYdhHhpiFes
BmwQvW2dyElNO1ee/zzQOZ0Hd3vM/vdbmZRg8zKoV5eQz/MA2jpxF9IkXEDtUMzZ
C5kqTMRXdoWR2ZP2HLZjUstRX7d9BP/8oeg+2lMq9UlULWrYyVaD85dEAmCt9mQz
TqtAMmtp5IXalT+vKwhMu0MwUfyXyvl7ery4kxfmFtEeJQyxrdK2gihDxr0ndxBH
mowlIoBiGYMhUr/aF0lM00blpRAUoGv1rO2DUWh8TXzRJq6FJ4pzo9XRwR0dP7A6
VBlBdhX2SlE9XJOjLq9ppV5Vr9u6ZauwEI6kBLoagjWOF3t34Gnvel8zO+H5fhhq
UKMOAbC0SYmLVIPNUauHYoKiJlWb+Dlr96NsPm0sYBbB047hr0evBUmkJNrulI3N
uQGNBGkpk+gBDADZu1B0itmBIGzxjGmUjK9UagW3HNLNVX3jn2Jwe7yl3vL/d3Fk
qBRydky32P4whbUSiJN71Tze9l+WOxeXGa0orKPPha/oQtlnmqyM6WBfOmvjSQkn
xCrS131SYjFb5dcQXoqTtUk8Wu4qoMHu/Mi9jtx8GrrENWzR7DFG2MDWwcM4TmUk
zs6azU/jaQX/YrGmYU3vB/zpKEZSo89pJ/S2FQ/6Fr+nnh7El3thNJxLRH40UlZs
FIpfm95Q+wtC224wckro/Xf/+t7oBssSqCZEkCItN+AlDpm13h161dypVRpC1M5Y
IITHq92gQSAKFhwSauaPtJ++oQNRgz5vJhCg78XfyBGpwDS6P0NM5RVEz6LMhQzI
4LziGmYH/iWj0pv45Uze7hOZaWhBPlPWFixJ97nl3soxA7hV1MLt/Ohy5jxGC8U3
CrtEjyAIQxCmEUF0vPa7X1KB+FxCpJ8mYXBZ5w1DwBN7qsOnnHZKFcaPGW8r1am6
Ab25ee028mua9RkAEQEAAYkBtgQYAQoAIBYhBLYCLMEHU5tANqkPrqutE80n+eaG
BQJpKZPoAhsMAAoJEKutE80n+eaGCRgMAJiG7Q8oF6oMkn6Xh7kXVH2yF4CKN9/j
/qtImK/ikn14+/QNYpUbF4kIGadeCVgpKZZ+R9QLXTW7WQV4hgoOW3yiET3FTEBc
YoxDxegwlk9+gPiOMJ0+9R43IUs+jFrra1jcUpSg+1Nv2IijdPwape3HyPhYgDmZ
VxORrNtqBCkhtLpJO5VTiThAny+rNBHk1t1vQg4tEkCLGcOD8bsdxhACZnM0DVYY
rA6afeDnm7CTfVtc3QFAi2+ltYcDIrMxMF0b1VASlbU14TE6ep2Ic30ScpDJL8De
skhWi2/0v2WwhbmdGzfv5K5V3Z6NtoB3OUaHzKqzgEQeqjudgRaL440UsDtEFRHE
vh6kVR0MPTFjg+8khUjjrSYUzqiiK6iJDxn+m0XJvHzQTeCHQlgRrPj9HGp7isyK
o9Wf8fVvnksR8xc/2NcJwp0fo7ULwdADjgMtPUR155ukI8Xt9Aws+5OsYrIxBMMc
DTh6aJal6iGcG4aXbTzwIFXHW9WipS5B4A==
=rrxA
-----END PGP PUBLIC KEY BLOCK-----
```

Cette clef peut ensuite être jointe à des messages électroniques ou bien déposée sur un serveur de clefs tel http://www.keyserver.net.

Signer un message

Créez maintenant un message à signer :

```
root@debian12:~# vi message.txt

root@debian12:~# cat message.txt
# ~/message.txt
Ceci est un message de test pour GnuPG
```

Pour signer ce message en format binaire, il convient d'utiliser la commande suivante :

Important - Entrez votre Passphrase quand gpg vous la demande.

```
root@debian12:~# gpg --default-key ITTRAINING --detach-sign message.txt
gpg: using "ITTRAINING" as default secret key for signing

root@debian12:~# ls -l | grep message
-rw-r--r-- 1 root    root         55 Nov 28 13:28 message.txt
-rw-r--r-- 1 root    root        461 Nov 28 13:28 message.txt.sig

root@debian12:~# cat message.txt.sig

4!,S@6'i)infos@ittraining.team
        '|
                M<~,#5@^0F6,ɱl`6fA(897>"Bj\
N!M.=F[3tC"%Fk@{6ƒbbEcc2'Smhlwm⌣)<AICA$U^*DPI0⊙)'p-4d{EiU\P`9}%L0RHG␣(-y$i'fQ̄CBH^c
```

```
;pQb bJeqKh7I/6FLc7kUi}/%3xeOv;i13Ĭ
```

Pour signer ce message en format ascii, il convient d'utiliser la commande suivante :

```
root@debian12:~#  gpg --default-key ITTRAINING --armor --detach-sign message.txt

gpg: using "ITTRAINING" as default secret key for signing

root@debian12:~# ls -l | grep message
-rw-r--r-- 1 root    root         55 Nov 28 13:28 message.txt
-rw-r--r-- 1 root    root        691 Nov 28 13:31 message.txt.asc
-rw-r--r-- 1 root    root        461 Nov 28 13:28 message.txt.sig

root@debian12:~# cat message.txt.asc
-----BEGIN PGP SIGNATURE-----

iQHKBAABCgA0FiEEtgIswQdTm0A2qQ+uq60TzSf55oYFAmkplgkWHGluZm9zQGl0
dHJhaW5pbmcudGVhbQAKCRCrrRPNJ/nmhiteC/kBXVFzjOQYIzdt4ALI37gCmd7x
Ml72F9permfoLc83fj+zZfigUxZe4DHx0TslVtFhntkg+7wF1H3MmUyKOhwCXFO4
ccyE9/DDJR384muuAP1q05bnXz0SHkTQaXVteCvAfHB8kzqQg04ePCBcIWK6YQVv
MBVm2O4pFDXWu1+0S69YnAelElZrzoqCaqprkQCcJvekBSV2NUNPLcwcBGq966+q
T9bGxPtW/5oTaPfFRzWwT/47ajpB66v3kEyGlc+gMpD4PFKZhLoEpgpByBWRWvhm
ZQsz2eby3oyv5KduA2emMnstQGCBq6zukYUpTgtzXB6jF1S71okiow11NsPra0m+
9CAcg3tiylnkUJEgsvfZ45uctN+2UqBNAXMgVsHGUhJtugfPzfwWpfC/BMMdLS46
g2nTttJexAGjjtw4Y0uFIYbNI0xXeH8ooh0AgQcXS5IOVH2zYs0GIdMxzRYFhRmc
nVRd/mHfQ21Mi1C5AnxDkqx6RpeH1maaLsPOFlo=
=2N1N
-----END PGP SIGNATURE-----
```

Pour signer ce message **dans le message lui-même** en format ascii, il convient d'utiliser la commande suivante :

```
root@debian12:~# gpg --default-key ITTRAINING --clearsign message.txt
gpg: using "ITTRAINING" as default secret key for signing
```

```
File 'message.txt.asc' exists. Overwrite? (y/N) y

root@debian12:~# ls -l | grep message
-rw-r--r-- 1 root    root         55 Nov 28 13:28 message.txt
-rw-r--r-- 1 root    root        795 Nov 28 13:33 message.txt.asc
-rw-r--r-- 1 root    root        461 Nov 28 13:28 message.txt.sig

root@debian12:~# cat message.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

# ~/message.txt
Ceci est un message de test pour GnuPG
-----BEGIN PGP SIGNATURE-----

iQHKBAEBCgA0FiEEtgIswQdTm0A2qQ+uq60TzSf55oYFAmkplpsWHGluZm9zQGl0
dHJhaW5pbmcudGVhbQAKCRCrrRPNJ/nmhopQC/wJYnC83AnA54x9O9FzHIQ0YVqO
QzcoDIIDMF71lizXIQ4DL2GCKhO3pelEWfyofUxd1sddT2qIHIsRTULaqHPRRjOU
9e/Wm9i65aUnQ/o5oDLSuHooi/r3HYPr8tHDhNSXT81a/QQ0BYVpMHVW/LREtw+L
qNPNlSZ4kzim3LyASyg8SYTGfft35S1S+7bjoY7LHfJUlGuSFRtlDMlhTbrrDqhI
S3TL6EeNFCdEVoxCPamsAKvuk4BV8Fe2rCjQqm/D6f0l+bgdnGFnd5y+nCiz0xXF
i7lQkZz+IjoTqwrjboL1mPWt6DMgiBX0IWas4kxKiBeZzyDZm6HGNeeMsqnReTcR
TPNM9FdDmpcxYzrRbYAoWTmdhiPdD/aXPEp+McyNIwwHgduUWuRGcifP2PVw6j0N
i3yOFw/rJEZMQnjwuL0GUSO3o8j1WMcSsTOHqsgu5FSKSjiIelNK2gzHTkuFiJ/7
scYMCHrlrUfErpkLD1H6eFhSA4StmdZPmAA5DBA=
=07RM
-----END PGP SIGNATURE-----
```

Pour vérifier la signature d'un message signé en mode ascii, il convient d'utiliser la commande :

```
root@debian12:~# gpg --verify message.txt.asc
gpg: Signature made Fri 28 Nov 2025 01:33:31 PM CET
gpg:                using RSA key B6022CC107539B4036A90FAEABAD13CD27F9E686
gpg:                issuer "infos@ittraining.team"
```

```
gpg: Good signature from "ITTRAINING (Test key) <infos@ittraining.team>" [ultimate]
gpg: WARNING: not a detached signature; file 'message.txt' was NOT verified!
```

Important - Pour vérifier la signature d'un message signé en mode ascii et produit en dehors du message lui-même, il convient d'utiliser la commande :

```
# gpg --verify message.txt.asc message.txt
```

Chiffrer un message

Pour chiffrer un message, il faut disposer de la clef publique du destinataire du message. Ce dernier utilisera ensuite sa clef privée pour déchiffrer le message. Il convient de préciser le destinataire du message, ou plus précisément la clef publique à utiliser, lors d'un chiffrement :

```
  gpg --recipient <destinataire> --encrypt <message>
```

- <*destinataire*> représente toute information permettant de distinguer sans ambigüité une clef publique dans votre trousseau. Cette information peut-être le nom ou l'adresse email associé à la clef publique que vous voulez utiliser,
- <*message*> représente le message à chiffrer.

Par exemple pour chiffrer un message en mode binaire, il convient de saisir la commande suivante :

```
root@debian12:~# gpg --recipient ITTRAINING --encrypt message.txt

root@debian12:~# ls -l | grep message
-rw-r--r-- 1 root    root         55 Nov 28 13:28 message.txt
-rw-r--r-- 1 root    root        795 Nov 28 13:33 message.txt.asc
-rw-r--r-- 1 root    root        510 Nov 28 13:35 message.txt.gpg
-rw-r--r-- 1 root    root        461 Nov 28 13:28 message.txt.sig
```

```
root@debian12:~# cat message.txt.gpg
əS4
   t>25H@H:x+H  n+B.^2Z4´~~~sd27J    ~:W^}ĀzmI]/e=kS□n9tSQ=Cj5{Q
                                                                  #0"  @qw,\[B$(:BU刮9^8_Qa
                                                                                          e^4,鹭
k/Pj#y:j~PU(}K#tT-AZu  1z!%]t{
5?"a5Vdnn$\V#(ctWhce6/-qThy^ъmk-$$Dn
^&Mw.*□YrJɻr}o/Q[X_)Wr?U0gMV,">9Croot@debian12:~#
```

Et pour chiffrer un message en mode ascii, il convient de saisir la commande suivante :

```
root@debian12:~# gpg --recipient ITTRAINING --armor --encrypt message.txt
File 'message.txt.asc' exists. Overwrite? (y/N) y

root@debian12:~# ls -l | grep message
-rw-r--r-- 1 root    root         55 Nov 28 13:28 message.txt
-rw-r--r-- 1 root    root        752 Nov 28 13:38 message.txt.asc
-rw-r--r-- 1 root    root        510 Nov 28 13:36 message.txt.gpg
-rw-r--r-- 1 root    root        461 Nov 28 13:28 message.txt.sig

root@debian12:~# cat message.txt.asc
-----BEGIN PGP MESSAGE-----

hQGMA9ObUzTZ86GvAQwAxAGoxKHBMh53+bPqHs338vhWjxAZs19aItMm+CM2pKu5
i4euJ36o+oVfbVTcqVNh4Q8I72QTOYTRTeXRWaUtVjp055A07BtlxGPbps9GXkVv
1faIs8viwHF1FHDC3Iz/SSf7u7tSF6sxbU6B4c/fHPneKjUxesH0XmKjduEdpDPj
ylZxPngzTOR4RDbMbkZ5OvJKXNrOOiHUSDOMmI9o9nu7sxnMiPP3NUqilN/0NtFz
xveV/mqDOEHT7Q1IHR+kN0e7WHqCl4NOvh0BXhL5u+9m18pgRe6ZvC6E94e+yPxZ
21umlYbcIPtw87+h9twBj7vK5FUi7j8C5QWoDoM4XERf/v21A1fTlx2cDn8E1lOC
dTaZlgTtxM3CY7/iWlG7TXWfUQfdZWhf/Mz1bHguljeOqVMGTI/zCElDnS3Ezzhg
bIlrpLbs6yqtjant1bkZ5PuNhB1bRNxrSKZUZKdEMeY5M+C0GjKskjN84+OqgjRV
39GkfPDwqw/eBj0ldvc60m0B5H2ZIIRApWl4j9RGhe2ZbvS+ZmI2vSf75RPDcNyH
PKDwHyTKCGN3NQKOIrw9LbIXehYEdrdFwGizGUXba1EWwVs+qMsmFueoQfcvA19N
CJ2HPuO3qAVIDpYyX+vwdKQASbb8AMQIIksoQv7i
```

```
=481C
-----END PGP MESSAGE-----
```

Pour décrypter un message il convient d'utiliser la commande suivante :

```
root@debian12:~# gpg --decrypt message.txt.asc
gpg: encrypted with 3072-bit RSA key, ID D39B5334D9F3A1AF, created 2025-11-28
      "ITTRAINING (Test key) <infos@ittraining.team>"
# ~/message.txt
Ceci est un message de test pour GnuPG
```

LAB #3 - Mise en place de SSH et SCP

3.1 - Introduction

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**, **RSA**, **ecdsa** ou **ed25519**. 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**, **ecdsa** ou **ed25519**,
- **RhostsRSA**,
- Par **mot de passe**.

L'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.

L'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.

3.2 - Configuration du Serveur

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

```
root@debian12:~# cat /etc/ssh/sshd_config

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

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

# 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 prohibit-password
#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)
KbdInteractiveAuthentication 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 KbdInteractiveAuthentication and
# PasswordAuthentication.  Depending on your PAM configuration,
# PAM authentication via KbdInteractiveAuthentication may bypass
# the setting of "PermitRootLogin prohibit-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and KbdInteractiveAuthentication 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 /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@debian12:~# cd /tmp ; grep -E -v '^(#|$)'  /etc/ssh/sshd_config > sshd_config

root@debian12:/tmp# cat sshd_config
Include /etc/ssh/sshd_config.d/*.conf
KbdInteractiveAuthentication 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@debian12:/tmp# cat sshd_config
Include /etc/ssh/sshd_config.d/*.conf
KbdInteractiveAuthentication no
UsePAM yes
PrintMotd no
AcceptEnv LANG LC_*
```

```
AllowGroups adm
HostbasedAuthentication no
IgnoreRhosts yes
LoginGraceTime 60
LogLevel INFO
PermitEmptyPasswords no
PermitRootLogin no
PrintLastLog yes
Protocol 2
StrictModes yes
X11Forwarding no
Subsystem       sftp    /usr/lib/openssh/sftp-server
```

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

```
root@debian12:/tmp# groups trainee
trainee : trainee cdrom floppy audio dip video plugdev netdev lpadmin scanner vboxusers

root@debian12:/tmp# usermod -a -G adm trainee

root@debian12:/tmp# groups trainee
trainee : trainee adm cdrom floppy audio dip video plugdev netdev lpadmin scanner vboxusers
```

Renommez le fichier **/etc/ssh/sshd_config** en **/etc/ssh/sshd_config.old** puis copiez le fichier **/tmp/sshd_config** vers **/etc/ssh/** :

```
root@debian12:/tmp# mv /etc/ssh/sshd_config /etc/ssh/sshd_config.old

root@debian12:/tmp# cp sshd_config /etc/ssh
```

Redémarrez ensuite le serveur ssh :

```
root@debian12:/tmp# systemctl restart ssh

root@debian12:/tmp# systemctl status ssh
```

```
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-11-28 15:18:55 CET; 7s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 10882 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 10883 (sshd)
      Tasks: 1 (limit: 19123)
     Memory: 1.4M
        CPU: 28ms
     CGroup: /system.slice/ssh.service
             └─10883 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Nov 28 15:18:55 debian12 systemd[1]: Starting ssh.service - OpenBSD Secure Shell server...
Nov 28 15:18:55 debian12 sshd[10883]: Server listening on 0.0.0.0 port 22.
Nov 28 15:18:55 debian12 sshd[10883]: Server listening on :: port 22.
Nov 28 15:18:55 debian12 systemd[1]: Started ssh.service - OpenBSD Secure Shell server.
```

Pour générer les clefs sur le serveur saisissez la commande suivante en tant que **root**:

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

```
root@debian12:/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:mP/fGCXEcyZQ+afGDrQK4S1TYVIw3pnBre25MPU2b3g root@debian12
```

```
The key's randomart image is:
+---[DSA 1024]----+
|        o++o..   |
|       ..o+*o    |
|        .o++=.o  |
|       o. o.+=. .|
|      o.S+ +.=.o |
|       .= + =oB  |
|        .+ +.* + |
|         .. .+o E|
|           ..o .o |
+----[SHA256]-----+
```

Important - Le chemin à indiquer pour le fichier est **/etc/ssh/ssh_host_dsa_key**. De la même façon, il est possible de générer les clefs au format **RSA**, **ECDSA** et **ED25519**.

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

```
root@debian12:/tmp# ls /etc/ssh
moduli      ssh_config.d  sshd_config.d    ssh_host_dsa_key      ssh_host_ecdsa_key      ssh_host_ed25519_key
ssh_host_rsa_key
ssh_config  sshd_config   sshd_config.old  ssh_host_dsa_key.pub  ssh_host_ecdsa_key.pub  ssh_host_ed25519_key.pub
ssh_host_rsa_key.pub
```

Re-démarrez ensuite le service sshd :

```
root@debian12:/tmp# systemctl restart ssh
```

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

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

```
root@debian12:/tmp# exit
logout

trainee@debian12:~$ 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:
SHA256:97XNFS0p/j8IaR7sByO1jVTE3sP23l/qDRUwU9TFRpw trainee@debian12
The key's randomart image is:
+---[DSA 1024]----+
|            o=o**|
|             o+E*|
|            + =+.|
|           + o *o|
|        S = * o =|
|         o % + =o|
|          = * =.=|
|           o o *+|
|            ..o *|
+----[SHA256]-----+

trainee@debian12:~$ 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:p2xPZdoPCICy/D5x0g+nHilsV6Ar4UMwmHeRnHDnBQw trainee@debian12
The key's randomart image is:
+---[RSA 3072]----+
|  .oE=o..        |
|.. .+=..         |
|= o o +          |
| = + . o         |
|  = ..  S . o    |
| o +o.++.+ *     |
|  + *++=+ + o    |
|   =.o.o.o   o   |
|    .o.   .   .  |
+----[SHA256]-----+

trainee@debian12:~$ 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:zA3muePCJZ7SCIvuKe+X5veMqWpP0UTr6HAVv+LPBCo trainee@debian12
The key's randomart image is:
+---[ECDSA 256]---+
|      o          |
|     . +         |
|      + +        |
|     * = =       |
```

```
|  . + = S .      |
|  .+ +.o..       |
| .Eo+*.++        |
|o +o* BO .       |
|+B+*++oo*        |
+----[SHA256]-----+

trainee@debian12:~$ 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:f5GzguZoS5SN2EQDyj7zAqES+7PDOBSJRg+lD+vpn3Q trainee@debian12
The key's randomart image is:
+--[ED25519 256]--+
|  ....o          |
| +.. . .         |
|=o*   .          |
|oO+. + +     .   |
|*.=.. = S   +    |
|o+.+ .   o   +   |
|.=+..E. o o o    |
|+ ++o..+   o     |
| oo+ .o..        |
+----[SHA256]-----+
```

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

3.3 - Utilisation

La commande ssh prend la forme suivante:

```
ssh -l nom_de_compte numero_ip (nom_de_machine)
```

En saisissant cette commande sur votre propre machine, vous obtiendrez un résultat similaire à celle-ci :

```
trainee@debian12:~$ su -
Password: fenestros
root@debian12:~#

root@debian12:~# ssh -l trainee localhost
The authenticity of host 'localhost (::1)' can't be established.
ED25519 key fingerprint is SHA256:SS9CpX7JFHIB54TWQWTtyswnwaXZ/Y8Kvr6dxPtisgE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
trainee@localhost's password:
Linux debian12 6.1.0-41-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.158-1 (2025-11-09) 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: Thu Nov 27 17:21:12 2025 from 10.0.2.1
```

3.4 - Mise en place des clefs

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

En saisissant cette commande, vous obtiendrez une fenêtre similaire à celle-ci :

```
root@debian12:~# exit
logout

trainee@debian12:~$ ssh -l trainee 127.0.0.1
trainee@127.0.0.1's password: trainee
Linux debian12 6.1.0-41-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.158-1 (2025-11-09) 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: Fri Nov 28 15:31:58 2025 from 10.0.2.46

trainee@debian12:~$ ls -la | grep .ssh
drwx------   2 trainee trainee 4096 Nov 28 15:29 .ssh

trainee@debian12:~$ exit
logout
Connection to 127.0.0.1 closed.
```

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** du 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@debian12:~$ scp .ssh/id_ecdsa.pub trainee@127.0.0.1:/home/trainee/.ssh/authorized_keys
trainee@127.0.0.1's password:
id_ecdsa.pub
100%  178   399.6KB/s   00:00
```

Connectez-vous via ssh et insérer les clefs publiques restantes dans le fichier .ssh/authorized_keys :

```
trainee@debian12:~$ ssh -l trainee localhost
Linux debian12 6.1.0-41-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.158-1 (2025-11-09) 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: Fri Nov 28 15:57:32 2025 from 127.0.0.1

trainee@debian12:~$ cat .ssh/id_rsa.pub >> .ssh/authorized_keys

trainee@debian12:~$ cat .ssh/id_dsa.pub >> .ssh/authorized_keys

trainee@debian12:~$ cat .ssh/id_ed25519.pub >> .ssh/authorized_keys

trainee@debian12:~$ cat .ssh/authorized_keys
ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBBwIXLMYJtWVgqSTqJfsFjS2ubtP+mHkC7XRU1rBWTBLYoJp8VOwtpKK1NTFS
aeJhnjCvFuhYm8egqHI0gp3oBA= trainee@debian12
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDcV1OmeHZlWhLumCozdg7snul5MrkzhZjiUKWNJmGBBOauOq2CgMGYWYIhjyeVeaeiezq+wSe0jO09p8a1E
bvYUibJsY9FOi8JcXTpxvlcqpuawudErjSsOf6JaIlbGpwDIMzmjeubfQx9jMZzfQIW9zfTDUi2tGquR/1TQPocGmmI9JFtoMqylk4+KsIHpriMBE
VD/aKiGr1wjLHR1v3er8P1o8YOd+9tZXouJxWLHIA1swrnpXt8doVWYGx0E9jWsScqI4JPTy8kb2hfWeZWqONIqgJty0kOeTt6IynOauf9fSjxOG9
Y4jyTV/OvTrBrmOCabjTIVMaIOWjLIPjNKJ8+eLnv+FuKUCc9mrAjbkXaP8T3JF43ofaBgSk09S3yo6wI3XI8ZlgROLDt7xxbFKosK/tYe8uOkpla
dNQHq3+EhAmzsfZJIstY0tNeK7aQ2KlsH6hZkLpqHsoKdzsLhRpMuPnV7wNlrikrbcvousDdQBpnzhNaXcm2BW+jmOs= trainee@debian12
```

```
ssh-dss
AAAAB3NzaC1kc3MAAACBAMbFW6WkZFJ/ueao6xjxBb+wbbIhIyiH/GM6W/7ayiNmCfFlYCxw9nNAVghYJYNcP2hB+l0qZ0JYrOMUX82sXAvtYbbO1
xSZWNTTrbYEyR+o2ZWlgYBDz0DYj++dLCl0zCGHOfRq2VHtCx93mCxCiw5NXNflxOuy2WoY7hawxV/RAAAAFQDURMf0f29i5or5GrE4JWlFVZbiIw
AAAIBqGpwpz+uQTKpJmgiisVA1t/0MRP2YstUVnU29Nr/C7EJ3bom3wx4At/bbQEfc7iVf0pwRUSJhdTy4t2vnFWghoeoj2ALUA4QVarVr7m0TY7m
8bYgAlPVBx674DDmHfUh99PTwYcrcJUJMpVd+mwVccnXGigpHhs84QvD6csmxCwAAAIAEtC/YVdYAxFUd+ScG0zrBSsPHA/cigCF5cywNeu+OZqrK
2y6yiJwp9T9bCL2nvzTWeTg/2Ev7v7Zh2oNKE5/mUoTDDn22omefi9Ecxl9W6Pyi8R6PxV+8mXtjHKPVNEQyqA8tAG1VN33MMgWz5HUEdwdUrOcmr
kpyXNsRzvgD6w== trainee@debian12
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILP736HSOOTZBe6+CHy8PINQ4gb9kKB4xj0JWWZy1Wgl trainee@debian12
```

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

Options de la commande

Les options de cette commande sont :

```
trainee@debian12:~$ 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 [argument ...]]
```

3.5 - 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_local> -L<port-local>:<hôte_distant>:<port_distant>
```

Dans votre cas, vous allez créer un tunnel entre Debian 12 et CentOS 8 entre le port 15023 et le port 23 :

```
trainee@debian12:~$ su -
Password: fenestros

root@debian12:~# ssh -N -f trainee@localhost -L15023:10.0.2.45:23
```

Installez maintenant le client et le serveur telnet dans la VM CentOS 8:

```
root@debian12:~# ssh -l trainee 10.0.2.45
The authenticity of host '10.0.2.45 (10.0.2.45)' can't be established.
ED25519 key fingerprint is SHA256:YRt3r4qD/pZ8PoJ2irS3bH2miZj6/O3/kS4CBlcbz84.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.45' (ED25519) to the list of known hosts.
trainee@10.0.2.45's password: trainee
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Thu Nov  6 04:13:03 2025 from 10.0.2.45

[trainee@centos8 ~]$ su -
Password: fenestros

[root@centos8 ~]# dnf install telnet-server
```

Telnet n'est ni démarré ni activé. Il convient donc de le démarrer et de l'activer :

```
[root@centos8 ~]# systemctl status telnet.socket
● telnet.socket - Telnet Server Activation Socket
   Loaded: loaded (/usr/lib/systemd/system/telnet.socket; disabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: man:telnetd(8)
   Listen: [::]:23 (Stream)
 Accepted: 0; Connected: 0;

[root@centos8 ~]# systemctl start telnet.socket

[root@centos8 ~]# systemctl status telnet.socket
● telnet.socket - Telnet Server Activation Socket
   Loaded: loaded (/usr/lib/systemd/system/telnet.socket; disabled; vendor preset: disabled)
   Active: active (listening) since Fri 2025-11-28 09:42:52 EST; 2s ago
     Docs: man:telnetd(8)
   Listen: [::]:23 (Stream)
 Accepted: 0; Connected: 0;
   CGroup: /system.slice/telnet.socket

Nov 28 09:42:52 centos8.ittraining.loc systemd[1]: Listening on Telnet Server Activation Socket.

[root@centos8 ~]# systemctl enable telnet.socket
Created symlink /etc/systemd/system/sockets.target.wants/telnet.socket → /usr/lib/systemd/system/telnet.socket.

[root@centos8 ~]# systemctl status telnet.socket
● telnet.socket - Telnet Server Activation Socket
   Loaded: loaded (/usr/lib/systemd/system/telnet.socket; enabled; vendor preset: disabled)
   Active: active (listening) since Fri 2025-11-28 09:42:52 EST; 43s ago
     Docs: man:telnetd(8)
   Listen: [::]:23 (Stream)
 Accepted: 0; Connected: 0;
   CGroup: /system.slice/telnet.socket
```

```
Nov 28 09:42:52 centos8.ittraining.loc systemd[1]: Listening on Telnet Server Activation Socket.
```

Arrêtez le service firewalld :

```
[root@centos8 ~]# systemctl stop firewalld

[root@centos8 ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Connectez-vous ensuite via telnet sur CentOS 8 en utilisant le port 15023 de votre VM Debian 12 :

```
root@debian12:~# telnet localhost 15023
Trying ::1...
Connected to localhost.
Escape character is '^]'.

Kernel 4.18.0-348.7.1.el8_5.x86_64 on an x86_64
centos8 login: trainee
Password:
Last login: Fri Nov 28 09:45:22 from 10.0.2.46

[trainee@centos8 ~]$ pwd
/home/trainee

[trainee@centos8 ~]$ exit
logout
Connection closed by foreign host.
```

```
root@debian12:~#
```

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

3.6 - SCP

Introduction

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@debian12:~$ touch /home/trainee/scp_test
```

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

```
trainee@debian12:~$ scp trainee@127.0.0.1:/home/trainee/scp_test /tmp/scp_test

trainee@debian12:~$ ls /tmp/scp_test
/tmp/scp_test
```

LAB#4 - Mise en place d'un serveur OpenVPN

4.1 - Installation

Téléchargez le script d'installation **openvpn-install.sh** :

```
root@debian12:~# wget https://git.io/vpn -O openvpn-install.sh
--2025-12-10 15:51:01--  https://git.io/vpn
Resolving git.io (git.io)... 140.82.112.22
Connecting to git.io (git.io)|140.82.112.22|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://raw.github.com/Nyr/openvpn-install/master/openvpn-install.sh [following]
--2025-12-10 15:51:01--  https://raw.github.com/Nyr/openvpn-install/master/openvpn-install.sh
Resolving raw.github.com (raw.github.com)... 185.199.108.133, 185.199.109.133, 185.199.111.133, ...
Connecting to raw.github.com (raw.github.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://raw.githubusercontent.com/Nyr/openvpn-install/master/openvpn-install.sh [following]
--2025-12-10 15:51:02--  https://raw.githubusercontent.com/Nyr/openvpn-install/master/openvpn-install.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.110.133,
185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 24962 (24K) [text/plain]
Saving to: ‘openvpn-install.sh’

openvpn-install.sh
```

```
100%[=======================================================================================================
========>]  24.38K  --.-KB/s    in 0s

2025-12-10 15:51:02 (93.5 MB/s) - ‘openvpn-install.sh’ saved [24962/24962]
```

Consultez le script

```
root@debian12:~# cat openvpn-install.sh
#!/bin/bash
#
# https://github.com/Nyr/openvpn-install
#
# Copyright (c) 2013 Nyr. Released under the MIT License.


# Detect Debian users running the script with "sh" instead of bash
if readlink /proc/$$/exe | grep -q "dash"; then
        echo 'This installer needs to be run with "bash", not "sh".'
        exit
fi

# Discard stdin. Needed when running from a one-liner which includes a newline
read -N 999999 -t 0.001

# Detect OS
# $os_version variables aren't always in use, but are kept here for convenience
if grep -qs "ubuntu" /etc/os-release; then
        os="ubuntu"
        os_version=$(grep 'VERSION_ID' /etc/os-release | cut -d '"' -f 2 | tr -d '.')
        group_name="nogroup"
elif [[ -e /etc/debian_version ]]; then
        os="debian"
        os_version=$(grep -oE '[0-9]+' /etc/debian_version | head -1)
        group_name="nogroup"
```

```
elif [[ -e /etc/almalinux-release || -e /etc/rocky-release || -e /etc/centos-release ]]; then
        os="centos"
        os_version=$(grep -shoE '[0-9]+' /etc/almalinux-release /etc/rocky-release /etc/centos-release | head -1)
        group_name="nobody"
elif [[ -e /etc/fedora-release ]]; then
        os="fedora"
        os_version=$(grep -oE '[0-9]+' /etc/fedora-release | head -1)
        group_name="nobody"
else
        echo "This installer seems to be running on an unsupported distribution.
Supported distros are Ubuntu, Debian, AlmaLinux, Rocky Linux, CentOS and Fedora."
        exit
fi

if [[ "$os" == "ubuntu" && "$os_version" -lt 2204 ]]; then
        echo "Ubuntu 22.04 or higher is required to use this installer.
This version of Ubuntu is too old and unsupported."
        exit
fi

if [[ "$os" == "debian" ]]; then
        if grep -q '/sid' /etc/debian_version; then
                echo "Debian Testing and Debian Unstable are unsupported by this installer."
                exit
        fi
        if [[ "$os_version" -lt 11 ]]; then
                echo "Debian 11 or higher is required to use this installer.
This version of Debian is too old and unsupported."
                exit
        fi
fi

if [[ "$os" == "centos" && "$os_version" -lt 9 ]]; then
        os_name=$(sed 's/ release.*//' /etc/almalinux-release /etc/rocky-release /etc/centos-release 2>/dev/null
```

```
| head -1)
        echo "$os_name 9 or higher is required to use this installer.
This version of $os_name is too old and unsupported."
        exit
fi

# Detect environments where $PATH does not include the sbin directories
if ! grep -q sbin <<< "$PATH"; then
        echo '$PATH does not include sbin. Try using "su -" instead of "su".'
        exit
fi

if [[ "$EUID" -ne 0 ]]; then
        echo "This installer needs to be run with superuser privileges."
        exit
fi

if [[ ! -e /dev/net/tun ]] || ! ( exec 7<>/dev/net/tun ) 2>/dev/null; then
        echo "The system does not have the TUN device available.
TUN needs to be enabled before running this installer."
        exit
fi

# Store the absolute path of the directory where the script is located
script_dir="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"

if [[ ! -e /etc/openvpn/server/server.conf ]]; then
        # Detect some Debian minimal setups where neither wget nor curl are installed
        if ! hash wget 2>/dev/null && ! hash curl 2>/dev/null; then
                echo "Wget is required to use this installer."
                read -n1 -r -p "Press any key to install Wget and continue..."
                apt-get update
                apt-get install -y wget
        fi
```

```
        clear
        echo 'Welcome to this OpenVPN road warrior installer!'
        # If system has a single IPv4, it is selected automatically. Else, ask the user
        if [[ $(ip -4 addr | grep inet | grep -vEc '127(\.[0-9]{1,3}){3}') -eq 1 ]]; then
                ip=$(ip -4 addr | grep inet | grep -vE '127(\.[0-9]{1,3}){3}' | cut -d '/' -f 1 | grep -oE
'[0-9]{1,3}(\.[0-9]{1,3}){3}')
        else
                number_of_ip=$(ip -4 addr | grep inet | grep -vEc '127(\.[0-9]{1,3}){3}')
                echo
                echo "Which IPv4 address should be used?"
                ip -4 addr | grep inet | grep -vE '127(\.[0-9]{1,3}){3}' | cut -d '/' -f 1 | grep -oE
'[0-9]{1,3}(\.[0-9]{1,3}){3}' | nl -s ') '
                read -p "IPv4 address [1]: " ip_number
                until [[ -z "$ip_number" || "$ip_number" =~ ^[0-9]+$ && "$ip_number" -le "$number_of_ip" ]]; do
                        echo "$ip_number: invalid selection."
                        read -p "IPv4 address [1]: " ip_number
                done
                [[ -z "$ip_number" ]] && ip_number="1"
                ip=$(ip -4 addr | grep inet | grep -vE '127(\.[0-9]{1,3}){3}' | cut -d '/' -f 1 | grep -oE
'[0-9]{1,3}(\.[0-9]{1,3}){3}' | sed -n "$ip_number"p)
        fi
        # If $ip is a private IP address, the server must be behind NAT
        if echo "$ip" | grep -qE '^(10\.|172\.1[6789]\.|172\.2[0-9]\.|172\.3[01]\.|192\.168)'; then
                echo
                echo "This server is behind NAT. What is the public IPv4 address or hostname?"
                # Get public IP and sanitize with grep
                get_public_ip=$(grep -m 1 -oE '^[0-9]{1,3}(\.[0-9]{1,3}){3}$' <<< "$(wget -T 10 -t 1 -4qO-
"http://ip1.dynupdate.no-ip.com/" || curl -m 10 -4Ls "http://ip1.dynupdate.no-ip.com/")")
                read -p "Public IPv4 address / hostname [$get_public_ip]: " public_ip
                # If the checkip service is unavailable and user didn't provide input, ask again
                until [[ -n "$get_public_ip" || -n "$public_ip" ]]; do
                        echo "Invalid input."
                        read -p "Public IPv4 address / hostname: " public_ip
                done
```

```
                [[ -z "$public_ip" ]] && public_ip="$get_public_ip"
        fi
        # If system has a single IPv6, it is selected automatically
        if [[ $(ip -6 addr | grep -c 'inet6 [23]') -eq 1 ]]; then
                ip6=$(ip -6 addr | grep 'inet6 [23]' | cut -d '/' -f 1 | grep -oE '([0-9a-fA-F]{0,4}:){1,7}[0-9a-
fA-F]{0,4}')
        fi
        # If system has multiple IPv6, ask the user to select one
        if [[ $(ip -6 addr | grep -c 'inet6 [23]') -gt 1 ]]; then
                number_of_ip6=$(ip -6 addr | grep -c 'inet6 [23]')
                echo
                echo "Which IPv6 address should be used?"
                ip -6 addr | grep 'inet6 [23]' | cut -d '/' -f 1 | grep -oE '([0-9a-fA-F]{0,4}:){1,7}[0-9a-fA-
F]{0,4}' | nl -s ') '
                read -p "IPv6 address [1]: " ip6_number
                until [[ -z "$ip6_number" || "$ip6_number" =~ ^[0-9]+$ && "$ip6_number" -le "$number_of_ip6" ]];
do
                        echo "$ip6_number: invalid selection."
                        read -p "IPv6 address [1]: " ip6_number
                done
                [[ -z "$ip6_number" ]] && ip6_number="1"
                ip6=$(ip -6 addr | grep 'inet6 [23]' | cut -d '/' -f 1 | grep -oE '([0-9a-fA-F]{0,4}:){1,7}[0-9a-
fA-F]{0,4}' | sed -n "$ip6_number"p)
        fi
        echo
        echo "Which protocol should OpenVPN use?"
        echo "   1) UDP (recommended)"
        echo "   2) TCP"
        read -p "Protocol [1]: " protocol
        until [[ -z "$protocol" || "$protocol" =~ ^[12]$ ]]; do
                echo "$protocol: invalid selection."
                read -p "Protocol [1]: " protocol
        done
        case "$protocol" in
```

```
                1|"")
                protocol=udp
                ;;
                2)
                protocol=tcp
                ;;
        esac
        echo
        echo "What port should OpenVPN listen on?"
        read -p "Port [1194]: " port
        until [[ -z "$port" || "$port" =~ ^[0-9]+$ && "$port" -le 65535 ]]; do
                echo "$port: invalid port."
                read -p "Port [1194]: " port
        done
        [[ -z "$port" ]] && port="1194"
        echo
        echo "Select a DNS server for the clients:"
        echo "   1) Default system resolvers"
        echo "   2) Google"
        echo "   3) 1.1.1.1"
        echo "   4) OpenDNS"
        echo "   5) Quad9"
        echo "   6) Gcore"
        echo "   7) AdGuard"
        echo "   8) Specify custom resolvers"
        read -p "DNS server [1]: " dns
        until [[ -z "$dns" || "$dns" =~ ^[1-8]$ ]]; do
                echo "$dns: invalid selection."
                read -p "DNS server [1]: " dns
        done
        # If the user selected custom resolvers, we deal with that here
        if [[ "$dns" = "8" ]]; then
                echo
                until [[ -n "$custom_dns" ]]; do
```

```
                        echo "Enter DNS servers (one or more IPv4 addresses, separated by commas or spaces):"
                        read -p "DNS servers: " dns_input
                        # Convert comma delimited to space delimited
                        dns_input=$(echo "$dns_input" | tr ',' ' ')
                        # Validate and build custom DNS IP list
                        for dns_ip in $dns_input; do
                                if [[ "$dns_ip" =~ ^[0-9]{1,3}(\.[0-9]{1,3}){3}$ ]]; then
                                        if [[ -z "$custom_dns" ]]; then
                                                custom_dns="$dns_ip"
                                        else
                                                custom_dns="$custom_dns $dns_ip"
                                        fi
                                fi
                        done
                        if [ -z "$custom_dns" ]; then
                                echo "Invalid input."
                        fi
                done
        fi
        echo
        echo "Enter a name for the first client:"
        read -p "Name [client]: " unsanitized_client
        # Allow a limited set of characters to avoid conflicts
        client=$(sed 's/[^0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_-]/_/g' <<<
"$unsanitized_client")
        [[ -z "$client" ]] && client="client"
        echo
        echo "OpenVPN installation is ready to begin."
        # Install a firewall if firewalld or iptables are not already available
        if ! systemctl is-active --quiet firewalld.service && ! hash iptables 2>/dev/null; then
                if [[ "$os" == "centos" || "$os" == "fedora" ]]; then
                        firewall="firewalld"
                        # We don't want to silently enable firewalld, so we give a subtle warning
                        # If the user continues, firewalld will be installed and enabled during setup
```

```
                        echo "firewalld, which is required to manage routing tables, will also be installed."
                elif [[ "$os" == "debian" || "$os" == "ubuntu" ]]; then
                        # iptables is way less invasive than firewalld so no warning is given
                        firewall="iptables"
                fi
        fi
        read -n1 -r -p "Press any key to continue..."
        # If running inside a container, disable LimitNPROC to prevent conflicts
        if systemd-detect-virt -cq; then
                mkdir /etc/systemd/system/openvpn-server@server.service.d/ 2>/dev/null
                echo "[Service]
LimitNPROC=infinity" > /etc/systemd/system/openvpn-server@server.service.d/disable-limitnproc.conf
        fi
        if [[ "$os" = "debian" || "$os" = "ubuntu" ]]; then
                apt-get update
                apt-get install -y --no-install-recommends openvpn openssl ca-certificates $firewall
        elif [[ "$os" = "centos" ]]; then
                dnf install -y epel-release
                dnf install -y openvpn openssl ca-certificates tar $firewall
        else
                # Else, OS must be Fedora
                dnf install -y openvpn openssl ca-certificates tar $firewall
        fi
        # If firewalld was just installed, enable it
        if [[ "$firewall" == "firewalld" ]]; then
                systemctl enable --now firewalld.service
        fi
        # Get easy-rsa
        easy_rsa_url='https://github.com/OpenVPN/easy-rsa/releases/download/v3.2.4/EasyRSA-3.2.4.tgz'
        mkdir -p /etc/openvpn/server/easy-rsa/
        { wget -qO- "$easy_rsa_url" 2>/dev/null || curl -sL "$easy_rsa_url" ; } | tar xz -C
/etc/openvpn/server/easy-rsa/ --strip-components 1
        chown -R root:root /etc/openvpn/server/easy-rsa/
        cd /etc/openvpn/server/easy-rsa/
```

```
        # Create the PKI, set up the CA and create TLS key
        ./easyrsa --batch init-pki
        ./easyrsa --batch build-ca nopass
        ./easyrsa gen-tls-crypt-key
        # Create the DH parameters file using the predefined ffdhe2048 group
        echo '-----BEGIN DH PARAMETERS-----
MIIBCAKCAQEA//////////+t+FRYortKmq/cViAnPTzx2LnFg84tNpWp4TZBFGQz
+8yTnc4kmz75fS/jY2MMddj2gbICrsRhetPfHtXV/WVhJDP1H18GbtCFY2VVPe0a
87VXE15/V8k1mE8McODmi3fipona8+/och3xWKE2rec1MKzKT0g6eXq8CrGCsyT7
YdEIqUuyyOP7uWrat2DX9GgdT0Kj3jlN9K5W7edjcrsZCwenyO4KbXCeAvzhzffi
7MA0BM0oNC9hkXL+nOmFg/+OTxIy7vKBg8P+OxtMb61zO7X8vC7CIAXFjvGDfRaD
ssbzSibBsu/6iGtCOGEoXJf//////////wIBAg==
-----END DH PARAMETERS-----' > /etc/openvpn/server/dh.pem
        # Make easy-rsa aware of our external DH file (prevents a warning)
        ln -s /etc/openvpn/server/dh.pem pki/dh.pem
        # Create certificates and CRL
        ./easyrsa --batch --days=3650 build-server-full server nopass
        ./easyrsa --batch --days=3650 build-client-full "$client" nopass
        ./easyrsa --batch --days=3650 gen-crl
        # Move the stuff we need
        cp pki/ca.crt pki/private/ca.key pki/issued/server.crt pki/private/server.key pki/crl.pem
/etc/openvpn/server
        cp pki/private/easyrsa-tls.key /etc/openvpn/server/tc.key
        # CRL is read with each client connection, while OpenVPN is dropped to nobody
        chown nobody:"$group_name" /etc/openvpn/server/crl.pem
        # Without +x in the directory, OpenVPN can't run a stat() on the CRL file
        chmod o+x /etc/openvpn/server/
        # Generate server.conf
        echo "local $ip
port $port
proto $protocol
dev tun
ca ca.crt
cert server.crt
```

```
key server.key
dh dh.pem
auth SHA512
tls-crypt tc.key
topology subnet
server 10.8.0.0 255.255.255.0" > /etc/openvpn/server/server.conf
        # IPv6
        if [[ -z "$ip6" ]]; then
                echo 'push "redirect-gateway def1 bypass-dhcp"' >> /etc/openvpn/server/server.conf
        else
                echo 'server-ipv6 fddd:1194:1194:1194::/64' >> /etc/openvpn/server/server.conf
                echo 'push "redirect-gateway def1 ipv6 bypass-dhcp"' >> /etc/openvpn/server/server.conf
        fi
        echo 'ifconfig-pool-persist ipp.txt' >> /etc/openvpn/server/server.conf
        # DNS
        case "$dns" in
                1|"")
                        # Locate the proper resolv.conf
                        # Needed for systems running systemd-resolved
                        if grep '^nameserver' "/etc/resolv.conf" | grep -qv '127.0.0.53' ; then
                                resolv_conf="/etc/resolv.conf"
                        else
                                resolv_conf="/run/systemd/resolve/resolv.conf"
                        fi
                        # Obtain the resolvers from resolv.conf and use them for OpenVPN
                        grep -v '^#\|^;' "$resolv_conf" | grep '^nameserver' | grep -v '127.0.0.53' | grep -oE
'[0-9]{1,3}(\.[0-9]{1,3}){3}' | while read line; do
                                echo "push \"dhcp-option DNS $line\"" >> /etc/openvpn/server/server.conf
                        done
                ;;
                2)
                        echo 'push "dhcp-option DNS 8.8.8.8"' >> /etc/openvpn/server/server.conf
                        echo 'push "dhcp-option DNS 8.8.4.4"' >> /etc/openvpn/server/server.conf
                ;;
```

```
                3)
                        echo 'push "dhcp-option DNS 1.1.1.1"' >> /etc/openvpn/server/server.conf
                        echo 'push "dhcp-option DNS 1.0.0.1"' >> /etc/openvpn/server/server.conf
                ;;
                4)
                        echo 'push "dhcp-option DNS 208.67.222.222"' >> /etc/openvpn/server/server.conf
                        echo 'push "dhcp-option DNS 208.67.220.220"' >> /etc/openvpn/server/server.conf
                ;;
                5)
                        echo 'push "dhcp-option DNS 9.9.9.9"' >> /etc/openvpn/server/server.conf
                        echo 'push "dhcp-option DNS 149.112.112.112"' >> /etc/openvpn/server/server.conf
                ;;
                6)
                        echo 'push "dhcp-option DNS 95.85.95.85"' >> /etc/openvpn/server/server.conf
                        echo 'push "dhcp-option DNS 2.56.220.2"' >> /etc/openvpn/server/server.conf
                ;;
                7)
                        echo 'push "dhcp-option DNS 94.140.14.14"' >> /etc/openvpn/server/server.conf
                        echo 'push "dhcp-option DNS 94.140.15.15"' >> /etc/openvpn/server/server.conf
                ;;
                8)
                for dns_ip in $custom_dns; do
                        echo "push \"dhcp-option DNS $dns_ip\"" >> /etc/openvpn/server/server.conf
                done
                ;;
        esac
        echo 'push "block-outside-dns"' >> /etc/openvpn/server/server.conf
        echo "keepalive 10 120
user nobody
group $group_name
persist-key
persist-tun
verb 3
crl-verify crl.pem" >> /etc/openvpn/server/server.conf
```

```
        if [[ "$protocol" = "udp" ]]; then
                echo "explicit-exit-notify" >> /etc/openvpn/server/server.conf
        fi
        # Enable net.ipv4.ip_forward for the system
        echo 'net.ipv4.ip_forward=1' > /etc/sysctl.d/99-openvpn-forward.conf
        # Enable without waiting for a reboot or service restart
        echo 1 > /proc/sys/net/ipv4/ip_forward
        if [[ -n "$ip6" ]]; then
                # Enable net.ipv6.conf.all.forwarding for the system
                echo "net.ipv6.conf.all.forwarding=1" >> /etc/sysctl.d/99-openvpn-forward.conf
                # Enable without waiting for a reboot or service restart
                echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
        fi
        if systemctl is-active --quiet firewalld.service; then
                # Using both permanent and not permanent rules to avoid a firewalld
                # reload.
                # We don't use --add-service=openvpn because that would only work with
                # the default port and protocol.
                firewall-cmd --add-port="$port"/"$protocol"
                firewall-cmd --zone=trusted --add-source=10.8.0.0/24
                firewall-cmd --permanent --add-port="$port"/"$protocol"
                firewall-cmd --permanent --zone=trusted --add-source=10.8.0.0/24
                # Set NAT for the VPN subnet
                firewall-cmd --direct --add-rule ipv4 nat POSTROUTING 0 -s 10.8.0.0/24 ! -d 10.8.0.0/24 -j SNAT -
-to "$ip"
                firewall-cmd --permanent --direct --add-rule ipv4 nat POSTROUTING 0 -s 10.8.0.0/24 ! -d
10.8.0.0/24 -j SNAT --to "$ip"
                if [[ -n "$ip6" ]]; then
                        firewall-cmd --zone=trusted --add-source=fddd:1194:1194:1194::/64
                        firewall-cmd --permanent --zone=trusted --add-source=fddd:1194:1194:1194::/64
                        firewall-cmd --direct --add-rule ipv6 nat POSTROUTING 0 -s fddd:1194:1194:1194::/64 ! -d
fddd:1194:1194:1194::/64 -j SNAT --to "$ip6"
                        firewall-cmd --permanent --direct --add-rule ipv6 nat POSTROUTING 0 -s
fddd:1194:1194:1194::/64 ! -d fddd:1194:1194:1194::/64 -j SNAT --to "$ip6"
```

```
                fi
        else
                # Create a service to set up persistent iptables rules
                iptables_path=$(command -v iptables)
                ip6tables_path=$(command -v ip6tables)
                # nf_tables is not available as standard in OVZ kernels. So use iptables-legacy
                # if we are in OVZ, with a nf_tables backend and iptables-legacy is available.
                if [[ $(systemd-detect-virt) == "openvz" ]] && readlink -f "$(command -v iptables)" | grep -q
"nft" && hash iptables-legacy 2>/dev/null; then
                        iptables_path=$(command -v iptables-legacy)
                        ip6tables_path=$(command -v ip6tables-legacy)
                fi
                echo "[Unit]
After=network-online.target
Wants=network-online.target
[Service]
Type=oneshot
ExecStart=$iptables_path -w 5 -t nat -A POSTROUTING -s 10.8.0.0/24 ! -d 10.8.0.0/24 -j SNAT --to $ip
ExecStart=$iptables_path -w 5 -I INPUT -p $protocol --dport $port -j ACCEPT
ExecStart=$iptables_path -w 5 -I FORWARD -s 10.8.0.0/24 -j ACCEPT
ExecStart=$iptables_path -w 5 -I FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
ExecStop=$iptables_path -w 5 -t nat -D POSTROUTING -s 10.8.0.0/24 ! -d 10.8.0.0/24 -j SNAT --to $ip
ExecStop=$iptables_path -w 5 -D INPUT -p $protocol --dport $port -j ACCEPT
ExecStop=$iptables_path -w 5 -D FORWARD -s 10.8.0.0/24 -j ACCEPT
ExecStop=$iptables_path -w 5 -D FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT" >
/etc/systemd/system/openvpn-iptables.service
                if [[ -n "$ip6" ]]; then
                        echo "ExecStart=$ip6tables_path -w 5 -t nat -A POSTROUTING -s fddd:1194:1194:1194::/64 !
-d fddd:1194:1194:1194::/64 -j SNAT --to $ip6
ExecStart=$ip6tables_path -w 5 -I FORWARD -s fddd:1194:1194:1194::/64 -j ACCEPT
ExecStart=$ip6tables_path -w 5 -I FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
ExecStop=$ip6tables_path -w 5 -t nat -D POSTROUTING -s fddd:1194:1194:1194::/64 ! -d fddd:1194:1194:1194::/64 -j
SNAT --to $ip6
ExecStop=$ip6tables_path -w 5 -D FORWARD -s fddd:1194:1194:1194::/64 -j ACCEPT
```

```
ExecStop=$ip6tables_path -w 5 -D FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT" >>
/etc/systemd/system/openvpn-iptables.service
                fi
                echo "RemainAfterExit=yes
[Install]
WantedBy=multi-user.target" >> /etc/systemd/system/openvpn-iptables.service
                systemctl enable --now openvpn-iptables.service
        fi
        # If SELinux is enabled and a custom port was selected, we need this
        if sestatus 2>/dev/null | grep "Current mode" | grep -q "enforcing" && [[ "$port" != 1194 ]]; then
                # Install semanage if not already present
                if ! hash semanage 2>/dev/null; then
                                dnf install -y policycoreutils-python-utils
                fi
                semanage port -a -t openvpn_port_t -p "$protocol" "$port"
        fi
        # If the server is behind NAT, use the correct IP address
        [[ -n "$public_ip" ]] && ip="$public_ip"
        # client-common.txt is created so we have a template to add further users later
        echo "client
dev tun
proto $protocol
remote $ip $port
resolv-retry infinite
nobind
persist-key
persist-tun
remote-cert-tls server
auth SHA512
ignore-unknown-option block-outside-dns
verb 3" > /etc/openvpn/server/client-common.txt
        # Enable and start the OpenVPN service
        systemctl enable --now openvpn-server@server.service
        # Build the $client.ovpn file, stripping comments from easy-rsa in the process
```

```
        grep -vh '^#' /etc/openvpn/server/client-common.txt /etc/openvpn/server/easy-
rsa/pki/inline/private/"$client".inline > "$script_dir"/"$client".ovpn
        echo
        echo "Finished!"
        echo
        echo "The client configuration is available in:" "$script_dir"/"$client.ovpn"
        echo "New clients can be added by running this script again."
else
        clear
        echo "OpenVPN is already installed."
        echo
        echo "Select an option:"
        echo "   1) Add a new client"
        echo "   2) Revoke an existing client"
        echo "   3) Remove OpenVPN"
        echo "   4) Exit"
        read -p "Option: " option
        until [[ "$option" =~ ^[1-4]$ ]]; do
                echo "$option: invalid selection."
                read -p "Option: " option
        done
        case "$option" in
                1)
                        echo
                        echo "Provide a name for the client:"
                        read -p "Name: " unsanitized_client
                        client=$(sed 's/[^0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_-]/_/g'
<<< "$unsanitized_client")
                        while [[ -z "$client" || -e /etc/openvpn/server/easy-rsa/pki/issued/"$client".crt ]]; do
                                echo "$client: invalid name."
                                read -p "Name: " unsanitized_client
                                client=$(sed
's/[^0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_-]/_/g' <<< "$unsanitized_client")
                        done
```

```
                        cd /etc/openvpn/server/easy-rsa/
                        ./easyrsa --batch --days=3650 build-client-full "$client" nopass
                        # Build the $client.ovpn file, stripping comments from easy-rsa in the process
                        grep -vh '^#' /etc/openvpn/server/client-common.txt /etc/openvpn/server/easy-
rsa/pki/inline/private/"$client".inline > "$script_dir"/"$client".ovpn
                        echo
                        echo "$client added. Configuration available in:" "$script_dir"/"$client.ovpn"
                        exit
                ;;
                2)
                        # This option could be documented a bit better and maybe even be simplified
                        # ...but what can I say, I want some sleep too
                        number_of_clients=$(tail -n +2 /etc/openvpn/server/easy-rsa/pki/index.txt | grep -c "^V")
                        if [[ "$number_of_clients" = 0 ]]; then
                                echo
                                echo "There are no existing clients!"
                                exit
                        fi
                        echo
                        echo "Select the client to revoke:"
                        tail -n +2 /etc/openvpn/server/easy-rsa/pki/index.txt | grep "^V" | cut -d '=' -f 2 | nl
-s ') '
                        read -p "Client: " client_number
                        until [[ "$client_number" =~ ^[0-9]+$ && "$client_number" -le "$number_of_clients" ]]; do
                                echo "$client_number: invalid selection."
                                read -p "Client: " client_number
                        done
                        client=$(tail -n +2 /etc/openvpn/server/easy-rsa/pki/index.txt | grep "^V" | cut -d '=' -
f 2 | sed -n "$client_number"p)
                        echo
                        read -p "Confirm $client revocation? [y/N]: " revoke
                        until [[ "$revoke" =~ ^[yYnN]*$ ]]; do
                                echo "$revoke: invalid selection."
                                read -p "Confirm $client revocation? [y/N]: " revoke
```

```
                        done
                        if [[ "$revoke" =~ ^[yY]$ ]]; then
                                cd /etc/openvpn/server/easy-rsa/
                                ./easyrsa --batch revoke "$client"
                                ./easyrsa --batch --days=3650 gen-crl
                                rm -f /etc/openvpn/server/crl.pem
                                rm -f /etc/openvpn/server/easy-rsa/pki/reqs/"$client".req
                                rm -f /etc/openvpn/server/easy-rsa/pki/private/"$client".key
                                cp /etc/openvpn/server/easy-rsa/pki/crl.pem /etc/openvpn/server/crl.pem
                                # CRL is read with each client connection, when OpenVPN is dropped to nobody
                                chown nobody:"$group_name" /etc/openvpn/server/crl.pem
                                echo
                                echo "$client revoked!"
                        else
                                echo
                                echo "$client revocation aborted!"
                        fi
                        exit
                ;;
                3)
                        echo
                        read -p "Confirm OpenVPN removal? [y/N]: " remove
                        until [[ "$remove" =~ ^[yYnN]*$ ]]; do
                                echo "$remove: invalid selection."
                                read -p "Confirm OpenVPN removal? [y/N]: " remove
                        done
                        if [[ "$remove" =~ ^[yY]$ ]]; then
                                port=$(grep '^port ' /etc/openvpn/server/server.conf | cut -d " " -f 2)
                                protocol=$(grep '^proto ' /etc/openvpn/server/server.conf | cut -d " " -f 2)
                                if systemctl is-active --quiet firewalld.service; then
                                        ip=$(firewall-cmd --direct --get-rules ipv4 nat POSTROUTING | grep '\-s
10.8.0.0/24 '"'"'!'"'"' -d 10.8.0.0/24' | grep -oE '[^ ]+$')
                                        # Using both permanent and not permanent rules to avoid a firewalld
reload.
```

```
                                        firewall-cmd --remove-port="$port"/"$protocol"
                                        firewall-cmd --zone=trusted --remove-source=10.8.0.0/24
                                        firewall-cmd --permanent --remove-port="$port"/"$protocol"
                                        firewall-cmd --permanent --zone=trusted --remove-source=10.8.0.0/24
                                        firewall-cmd --direct --remove-rule ipv4 nat POSTROUTING 0 -s 10.8.0.0/24
! -d 10.8.0.0/24 -j SNAT --to "$ip"
                                        firewall-cmd --permanent --direct --remove-rule ipv4 nat POSTROUTING 0 -s
10.8.0.0/24 ! -d 10.8.0.0/24 -j SNAT --to "$ip"
                                        if grep -qs "server-ipv6" /etc/openvpn/server/server.conf; then
                                                ip6=$(firewall-cmd --direct --get-rules ipv6 nat POSTROUTING |
grep '\-s fddd:1194:1194:1194::/64 '"'"'!'"'"' -d fddd:1194:1194:1194::/64' | grep -oE '[^ ]+$')
                                                firewall-cmd --zone=trusted --remove-
source=fddd:1194:1194:1194::/64
                                                firewall-cmd --permanent --zone=trusted --remove-
source=fddd:1194:1194:1194::/64
                                                firewall-cmd --direct --remove-rule ipv6 nat POSTROUTING 0 -s
fddd:1194:1194:1194::/64 ! -d fddd:1194:1194:1194::/64 -j SNAT --to "$ip6"
                                                firewall-cmd --permanent --direct --remove-rule ipv6 nat
POSTROUTING 0 -s fddd:1194:1194:1194::/64 ! -d fddd:1194:1194:1194::/64 -j SNAT --to "$ip6"
                                        fi
                                else
                                        systemctl disable --now openvpn-iptables.service
                                        rm -f /etc/systemd/system/openvpn-iptables.service
                                fi
                                if sestatus 2>/dev/null | grep "Current mode" | grep -q "enforcing" && [[ "$port"
!= 1194 ]]; then
                                        semanage port -d -t openvpn_port_t -p "$protocol" "$port"
                                fi
                                systemctl disable --now openvpn-server@server.service
                                rm -f /etc/systemd/system/openvpn-server@server.service.d/disable-limitnproc.conf
                                rm -f /etc/sysctl.d/99-openvpn-forward.conf
                                if [[ "$os" = "debian" || "$os" = "ubuntu" ]]; then
                                        rm -rf /etc/openvpn/server
                                        apt-get remove --purge -y openvpn
```

```
                                else
                                        # Else, OS must be CentOS or Fedora
                                        dnf remove -y openvpn
                                        rm -rf /etc/openvpn/server
                                fi
                                echo
                                echo "OpenVPN removed!"
                        else
                                echo
                                echo "OpenVPN removal aborted!"
                        fi
                        exit
                ;;
                4)
                        exit
                ;;
        esac
fi
```

Rendez ce script exécutable :

```
root@debian12:~# chmod +x openvpn-install.sh
```

Exécitez le script sous bash :

```
root@debian12:~# bash openvpn-install.sh
Welcome to this OpenVPN road warrior installer!

This server is behind NAT. What is the public IPv4 address or hostname?
Public IPv4 address / hostname [167.114.159.174]: 10.0.2.46

Which protocol should OpenVPN use?
   1) UDP (recommended)
   2) TCP
```

```
Protocol [1]:

What port should OpenVPN listen on?
Port [1194]:

Select a DNS server for the clients:
   1) Default system resolvers
   2) Google
   3) 1.1.1.1
   4) OpenDNS
   5) Quad9
   6) Gcore
   7) AdGuard
   8) Specify custom resolvers
DNS server [1]: 2

Enter a name for the first client:
Name [client]: client

OpenVPN installation is ready to begin.
Press any key to continue...
Hit:1 http://deb.debian.org/debian bookworm InRelease
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:3 http://security.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Hit:4 http://download.virtualbox.org/virtualbox/debian bookworm InRelease
Get:5 http://security.debian.org/debian-security bookworm-security/main Sources [195 kB]
Get:6 http://security.debian.org/debian-security bookworm-security/main amd64 Packages [290 kB]
Get:7 http://security.debian.org/debian-security bookworm-security/main Translation-en [176 kB]
Fetched 764 kB in 1s (797 kB/s)
Reading package lists... Done
W: http://download.virtualbox.org/virtualbox/debian/dists/bookworm/InRelease: Key is stored in legacy trusted.gpg
keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
```

```
Reading state information... Done
openssl is already the newest version (3.0.17-1~deb12u3).
ca-certificates is already the newest version (20230311+deb12u1).
The following additional packages will be installed:
  libpkcs11-helper1
Suggested packages:
  resolvconf openvpn-dco-dkms openvpn-systemd-resolved
Recommended packages:
  easy-rsa
The following NEW packages will be installed:
  libpkcs11-helper1 openvpn
0 upgraded, 2 newly installed, 0 to remove and 7 not upgraded.
Need to get 703 kB of archives.
After this operation, 1,968 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bookworm/main amd64 libpkcs11-helper1 amd64 1.29.0-1 [51.2 kB]
Get:2 http://security.debian.org/debian-security bookworm-security/main amd64 openvpn amd64 2.6.3-1+deb12u4 [652
kB]
Fetched 703 kB in 0s (23.0 MB/s)
Preconfiguring packages ...
Selecting previously unselected package libpkcs11-helper1:amd64.
(Reading database ... 188849 files and directories currently installed.)
Preparing to unpack .../libpkcs11-helper1_1.29.0-1_amd64.deb ...
Unpacking libpkcs11-helper1:amd64 (1.29.0-1) ...
Selecting previously unselected package openvpn.
Preparing to unpack .../openvpn_2.6.3-1+deb12u4_amd64.deb ...
Unpacking openvpn (2.6.3-1+deb12u4) ...
Setting up libpkcs11-helper1:amd64 (1.29.0-1) ...
Setting up openvpn (2.6.3-1+deb12u4) ...
Created symlink /etc/systemd/system/multi-user.target.wants/openvpn.service →
/lib/systemd/system/openvpn.service.
Processing triggers for man-db (2.11.2-2) ...
Processing triggers for libc-bin (2.36-9+deb12u13) ...
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.metadata:kMDItemTextContentLanguage'
```

```
Notice
------
'init-pki' complete; you may now create a CA or requests.

Your newly created PKI dir is:
* /etc/openvpn/server/easy-rsa/pki

Using Easy-RSA configuration:
* undefined
.............+......++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*..+..+....+.....+...+...+..
.....+.....+.+.....+....+..+..............++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*..+.
..+.....+.++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.........+..........+..+...+....+........+...+....+......+..+............+...+......+.........++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++*...+....+..+............++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++*..+................+.....+....+........+.......+...+.....+......+....+...+........+.....
..+...+...........+.........+......+......+............+...+....+.....+....+......+...........+...+....+..+...+.+
.....+......+.+........+.+......+.............+..............+....+......+...+..............+...+...+.........
...+..+.+.....+.........+.+..+.......+.....+...+....+...+..............+......+......+...+..............+......
+..+...+..............+......+.+.....+...+....+.....+..........+......++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++

Notice
------
CA creation complete. Your new CA certificate is at:
* /etc/openvpn/server/easy-rsa/pki/ca.crt

Build-ca completed successfully.


Notice
------
TLS-CRYPT Key generated at:
* /etc/openvpn/server/easy-rsa/pki/private/easyrsa-tls.key
```

```
If this file is changed then it MUST be redistributed to ALL servers
AND clients, to be in effect. Do NOT change this existing file.

..+.....................+..+.+..+....+...+..........+......+.+...........++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++*.......+..+.+..+.......+.....+.+..++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++*.....................+..........+.....+.+......+...........+.........++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++
........+....+...........+...........+..+.........+.+......+..++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++*..+...+..+..........+.....+....++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*......+.+..+....+........+...+.........+.........+.........+.+......+...........+...+...............+.......+...
..+.........+....+.........+.....+.+............+........+......+.+.....+....+.....+..........+.....+.........+..
..+......+.................+.......+...........+.++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+
-----

Notice
------
Private-Key and Public-Certificate-Request files created.
Your files are:
* req: /etc/openvpn/server/easy-rsa/pki/reqs/server.req
* key: /etc/openvpn/server/easy-rsa/pki/private/server.key

Using configuration from /etc/openvpn/server/easy-rsa/pki/58efa3df/temp.02
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'server'
Certificate is to be certified until Dec  8 14:57:02 2035 GMT (3650 days)

Write out database with 1 new entries
Database updated

Notice
------
```

```
Inline file created:
* /etc/openvpn/server/easy-rsa/pki/inline/private/server.inline


Notice
------
Certificate created at:
* /etc/openvpn/server/easy-rsa/pki/issued/server.crt

....+...+........+......+......+....+...+......+.....+....+...+..+.......+..............+.......+.....+.......+..
.+..+......+.+...+..+.+.....+.........+.+...+..+....+.........+.....++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++*...+.........+.+........+...+....+...+..++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++*...........+.+...+..+....+........+...+.............+......+...+..+.........+.......+..+......+.+..
.+...+..+......+....+..+......+.........+....+..+................+...+..........+.....+....+......+......+.....+..
.+...+...................+...+..+.........+....+........+....+..................+......+..+...+.........+.......+
......+...+.................+..........+...+............+...+...........+...+...+....+.........+......+...+.....+
......+............+..............+.......+......+..+.......+......+...........+......+..........+...+.....+....
.....+......+.+......+........+......+...+.......+...+...+...+...........+...+.+......+...+..+...+.......+...+...
.....+...+.+......+..+............+...+.....+......+......+.......+..+.+.....+.......+..+...+............+...+..
....+..........+.........+...+..+......+.+...........+...+....+...........+.+.........................+.........
....+............+........+.+...+..+.+...+.....+..........+.....+.+.....+.........+.....................+.+...+..
....+...+..+...+............+.....+....+.....+.+..+......+..........+..+....+......+.........+...+..+.+...+.....
....+.....+.............................+...+.....+.+.....................+...+......+....+..+...................+.
.+...+....+............+...+........+...+....+.....+......+.+...+......+......+.........+.......................
........+............+......+..................+....+...+...+.........+......+.+.....+.+........+.......+.....+
.......+...........+......+.+.....+....+...+.....+..........+.....+.++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
..+....+...+.....+......+.+......+............+........+............+.+.....+.+.........+............+...+.......
....+...++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.+.+..+.......+.....+......+....+...+..
....+..+...+......+....+...........+...+.......+...+...........+..........+.....+.+..............+....+..+.......
..+.+...+..+....+.....+......+....+.............+.+.....++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++*..++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-----
```

```
Notice
------
Private-Key and Public-Certificate-Request files created.
Your files are:
* req: /etc/openvpn/server/easy-rsa/pki/reqs/client.req
* key: /etc/openvpn/server/easy-rsa/pki/private/client.key

Using configuration from /etc/openvpn/server/easy-rsa/pki/36a78479/temp.02
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'client'
Certificate is to be certified until Dec  8 14:57:03 2035 GMT (3650 days)

Write out database with 1 new entries
Database updated

Notice
------
Inline file created:
* /etc/openvpn/server/easy-rsa/pki/inline/private/client.inline


Notice
------
Certificate created at:
* /etc/openvpn/server/easy-rsa/pki/issued/client.crt

Using configuration from /etc/openvpn/server/easy-rsa/openssl-easyrsa.cnf

Notice
------
An updated CRL DER copy has been created:
* /etc/openvpn/server/easy-rsa/pki/crl.der
```

```
An updated CRL has been created:
* /etc/openvpn/server/easy-rsa/pki/crl.pem

IMPORTANT: When the CRL expires, an OpenVPN Server which uses a
CRL will reject ALL new connections, until the CRL is replaced.

success
success
success
success
success
success
Created symlink /etc/systemd/system/multi-user.target.wants/openvpn-server@server.service →
/lib/systemd/system/openvpn-server@.service.

Finished!

The client configuration is available in: /root/client.ovpn
New clients can be added by running this script again.
```

Copiez le fichier de configuration du client dans le répertoire **/tmp** :

```
root@debian12:~# mv client.ovpn /tmp
```

4.2 - Configuration du Serveur

Consultez le fichier de configuration du serveur :

```
root@debian12:~# ls /etc/openvpn/server

root@debian12:~#  ls
ca.crt  client-common.txt  dh.pem    ipp.txt      server.crt  tc.key
ca.key  crl.pem            easy-rsa  server.conf  server.key
```

```
root@debian12:~# cat /etc/openvpn/server/server.conf
local 10.0.2.46
port 1194
proto udp
dev tun
ca ca.crt
cert server.crt
key server.key
dh dh.pem
auth SHA512
tls-crypt tc.key
topology subnet
server 10.8.0.0 255.255.255.0
push "redirect-gateway def1 bypass-dhcp"
ifconfig-pool-persist ipp.txt
push "dhcp-option DNS 8.8.8.8"
push "dhcp-option DNS 8.8.4.4"
push "block-outside-dns"
keepalive 10 120
user nobody
group nogroup
persist-key
persist-tun
verb 3
crl-verify crl.pem
explicit-exit-notify
```

Voici l'explication détaillée de chaque directive dans ce fichier server.conf :

Adresses, Ports et Rôle

local 10.0.2.46

- **Rôle** : Spécifie l'adresse IP locale sur laquelle le serveur OpenVPN doit écouter les connexions entrantes.
- **Détail** : Le serveur se liera uniquement à l'interface réseau ayant l'adresse 10.0.2.46. Ceci est utile si la machine serveur a plusieurs adresses IP.

port 1194

- **Rôle** : Définit le port TCP/UDP sur lequel le serveur écoute.

proto udp

- **Rôle** : Définit le protocole de transport à utiliser (UDP est préféré pour ses meilleures performances).

dev tun

- **Rôle** : Utilise un périphérique de tunneling TUN pour créer le réseau virtuel de couche 3 (IP).

Sécurité et Infrastructure à Clé Publique (PKI)

Ces directives définissent les composants cryptographiques et les mesures de sécurité essentielles.

ca ca.crt

- **Rôle** : Chemin vers le certificat de l'Autorité de Certification (CA). Utilisé pour vérifier l'authenticité des certificats clients.

cert server.crt

- **Rôle** : Chemin vers le certificat public du serveur.

key server.key

- **Rôle** : Chemin vers la clé privée du serveur (doit rester secrète).

dh dh.pem

- **Rôle** : Chemin vers les paramètres Diffie-Hellman pour l'échange de clés (établissement d'une clé de session sécurisée).

auth SHA512

- **Rôle** : Définit l'algorithme de hachage (HMAC) utilisé pour l'authentification des paquets de données (SHA512).

tls-crypt tc.

- **Rôle** : Spécifie l'utilisation d'une clé pré-partagée symétrique (tc.key) pour chiffrer l'ensemble du canal de contrôle TLS (y compris la négociation initiale).
- **Utilité** : Protège contre les attaques DoS et obscurcit le trafic OpenVPN pour le rendre indétectable par la plupart des outils d'inspection réseau.

crl-verify crl.pem

- **Rôle** : Active la vérification des certificats clients par une Liste de Révocation de Certificats (CRL).
- **Détail** : Si un certificat client est révoqué (par exemple, si le client a perdu sa clé privée), l'entrée est ajoutée à ce fichier (crl.pem), et le serveur refusera toute connexion de ce client. C'est une mesure de sécurité cruciale.

Configuration du Réseau Virtuel (VPN)

Ces directives définissent comment le serveur gère le réseau interne du VPN.

topology subnet

- **Rôle** : Définit la topologie du réseau virtuel comme un sous-réseau IP routé.
- **Détail** : Avec cette topologie, les clients voient l'interface du serveur comme une passerelle et peuvent potentiellement se voir mutuellement (contrairement à la topologie p2p qui ne permet pas aux clients de communiquer directement).

server 10.8.0.0 255.255.255.0

- **Rôle** : Configure le pool d'adresses IP virtuel pour les clients.
- **Détail** : Le sous-réseau VPN sera 10.8.0.0/24. Le serveur se verra attribuer 10.8.0.1, et les clients recevront dynamiquement des adresses dans la plage 10.8.0.x.

ifconfig-pool-persist ipp.txt

- **Rôle** : Mémorise l'adresse IP virtuelle attribuée à chaque client dans un fichier (ipp.txt).
- **Utilité** : Lorsque le client se reconnecte, il reçoit la même adresse IP qu'il avait précédemment (adresses IP persistantes), ce qui facilite le suivi et les règles de pare-feu.

Directives "Push" (Propagées aux Clients)

Ces commandes sont envoyées (poussées) par le serveur aux clients pour configurer leur routage et DNS.

push "redirect-gateway def1 bypass-dhcp"

- **Rôle** : Indique au client de rediriger tout son trafic Internet (la passerelle par défaut) via le tunnel VPN. C'est le mode “VPN complet”.

Détail : def1 assure que les routes ne suppriment pas accidentellement les routes existantes, et bypass-dhcp est une option de contournement standard.

push "dhcp-option DNS 8.8.8.8" et push "dhcp-option DNS 8.8.4.4"

- **Rôle** : Envoie les adresses des serveurs DNS publics de Google aux clients.
- **Utilité** : Assure que les requêtes DNS passent par le tunnel, empêchant les fuites DNS.

push "block-outside-dns"

- **Rôle** : (Spécifique aux clients Windows) Ordonne au client de bloquer l'utilisation de tout serveur DNS configuré en dehors du tunnel VPN.
- **Utilité** : Mesure de sécurité très forte pour éviter les fuites DNS.

Optimisation et Maintenance

keepalive 10 120

- **Rôle** : Configure les paquets de maintien de connexion (keepalive).
- **Détail** : Ping du serveur toutes les 10 secondes ; si aucune réponse n'est reçue pendant 120 secondes, la connexion est considérée comme perdue.

user nobody et group nogroup

- **Rôle** : Mesure de sécurité. Après l'initialisation (qui nécessite des privilèges root), le serveur abandonne les privilèges et exécute le processus VPN sous un utilisateur et groupe non privilégiés pour minimiser les risques en cas de compromission.

persist-key et persist-tun

- **Rôle** : Empêche le rechargement des clés et la réinitialisation de l'interface TUN/TAP lors de la reconnexion, ce qui rend les reconnexions plus rapides et plus fluides.

verb 3

- **Rôle** : Définit le niveau de journalisation (verbosité) à 3, fournissant un bon équilibre entre informations et clarté.

explicit-exit-notify

- **Rôle** : Active l'envoi d'une notification explicite au pair (le client, dans ce cas) lorsque le serveur est sur le point de s'arrêter.
- **Utilité** : Permet aux clients de savoir immédiatement que la connexion a été fermée intentionnellement, plutôt que d'attendre l'expiration du délai d'attente.

4.3 - Configuration du client

Connectez-vous à votre VM **CentOS8_10.0.2.45_SSH**.

Installez le client OpenVPN :

```
[root@centos8 ~]# dnf install epel-release

[root@centos8 ~]# dnf install openvpn
```

Téléchargez le fichier de configuration du client à partir du serveur VPN :

```
[root@centos8 ~]# scp trainee@10.0.2.46:/tmp/client.ovpn .
The authenticity of host '10.0.2.46 (10.0.2.46)' can't be established.
ECDSA key fingerprint is SHA256:JFem/0UXFw0aDAOSfOS3vsOGsSDl1wPOza6ybTGO7/8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.46' (ECDSA) to the list of known hosts.
trainee@10.0.2.46's password:
client.ovpn                                   100% 8254    68.5KB/s   00:00
```

Consultez le contenu du fichier client :

```
[root@centos8 ~]# cat client.ovpn
client
dev tun
proto udp
remote 10.0.2.46 1194
resolv-retry infinite
nobind
persist-key
persist-tun
remote-cert-tls server
auth SHA512
ignore-unknown-option block-outside-dns
verb 3

<cert>
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            ee:e8:be:eb:a3:df:5f:1a:ef:e0:b4:87:51:5d:9b:70
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=Easy-RSA CA
        Validity
            Not Before: Dec 10 18:03:29 2025 GMT
            Not After : Dec  8 18:03:29 2035 GMT
        Subject: CN=client
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:aa:8a:50:bd:6b:d6:f0:56:13:48:58:57:3c:3d:
                    e9:ac:1f:57:8f:4c:06:69:49:33:5c:50:87:76:d9:
                    58:e2:78:42:20:13:76:67:cb:ad:99:01:b0:83:1e:
                    38:7b:cd:f5:4d:42:e7:fb:d8:94:8e:e3:e0:57:e6:
```

```
                    fb:51:c3:b6:03:a7:bb:cb:1e:c8:47:fc:6d:ee:df:
                    15:c8:3f:fe:3b:21:83:95:ce:86:cc:08:7f:b1:57:
                    f3:2d:a2:a8:f3:71:5b:cc:d2:c2:81:09:2d:e3:ff:
                    66:70:33:e9:60:c2:d6:a8:da:0b:c5:86:3a:ac:5a:
                    25:cb:ab:93:c5:3e:02:d1:3b:cb:2d:bd:33:e1:22:
                    9c:87:b2:8a:c9:36:41:2d:fa:29:e5:7c:ec:41:f9:
                    10:ed:25:71:af:e9:69:aa:28:d4:35:80:82:92:bc:
                    4a:46:45:01:4f:78:83:99:28:bc:f3:2d:f1:d2:0e:
                    16:10:a3:e3:ff:68:06:6d:c0:56:a5:c1:ce:fb:e3:
                    7b:1b:6a:70:84:19:dd:bb:3f:01:9d:01:b8:1b:b0:
                    3d:7c:0f:61:4a:59:65:e9:44:a4:3e:44:1a:c2:5b:
                    70:a0:aa:27:73:ed:94:79:7c:36:af:a3:03:5c:0b:
                    d8:55:89:01:0a:80:8d:99:ea:b4:d6:c1:44:7e:bc:
                    20:69
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Key Identifier:
                A8:A0:35:A2:3B:84:B7:B2:69:2D:8C:B0:6A:88:9C:49:CD:E5:FA:04
            X509v3 Authority Key Identifier:
                keyid:1A:2A:90:CF:03:7F:5E:FA:7E:7C:0C:A0:F3:89:83:9C:E7:6B:DA:26
                DirName:/CN=Easy-RSA CA
                serial:3C:CB:56:8C:9C:8E:F5:F8:E9:E1:73:FD:31:49:87:EF:79:1E:32:16
            X509v3 Extended Key Usage:
                TLS Web Client Authentication
            X509v3 Key Usage:
                Digital Signature
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        4c:a4:64:c8:5e:88:c2:bc:e0:82:17:5e:a1:49:72:0f:18:bb:
        e5:81:2a:de:23:07:46:f0:2e:74:8a:70:23:89:a3:f9:6e:29:
        4f:44:cd:ad:aa:ba:8d:b1:28:84:0e:73:02:87:6a:5c:84:da:
        08:06:17:03:f3:e4:c4:94:73:91:09:2d:50:99:7f:6c:59:ce:
```

```
        e4:b3:93:cc:80:d1:a3:e7:36:15:ce:18:8b:b4:84:a1:7d:8c:
        1d:6e:5d:e0:77:c0:21:e4:ef:85:c0:1e:c5:a6:ca:5d:eb:e2:
        4c:8a:dc:b0:12:ec:4c:04:7a:66:e8:00:78:a7:ca:4d:2a:00:
        2a:65:6c:c7:50:67:32:5a:8f:8e:52:0c:eb:5b:ee:ea:13:c7:
        9d:d3:6d:f6:66:ea:08:f5:e8:7c:7b:6f:fe:af:fe:fa:b5:d1:
        be:4b:1c:5c:3b:da:6d:90:92:b2:2a:5a:85:90:33:6e:11:39:
        ba:f5:b9:57:c6:11:f8:f9:b7:e0:4c:0e:af:a0:9a:df:d3:54:
        39:dd:b0:aa:03:36:e4:e7:a2:e9:64:b8:a2:ab:92:96:8b:16:
        a6:74:4b:40:bf:80:3a:32:32:7c:4b:b9:af:2f:dc:22:8a:43:
        ab:9f:b5:33:12:97:f7:cb:df:a0:e1:a8:b0:6e:f9:1b:61:f0:
        b1:f9:57:6c
-----BEGIN CERTIFICATE-----
MIIDVTCCAj2gAwIBAgIRAO7ovuuj318a7+C0h1Fdm3AwDQYJKoZIhvcNAQELBQAw
FjEUMBIGA1UEAwwLRWFzeS1SU0EgQ0EwHhcNMjUxMjEwMTgwMzI5WhcNMzUxMjA4
MTgwMzI5WjARMQ8wDQYDVQQDDAZjbGllbnQwggEiMA0GCSqGSIb3DQEBAQUAA4IB
DwAwggEKAoIBAQCqilC9a9bwVhNIWFc8PemsH1ePTAZpSTNcUId22VjieEIgE3Zn
y62ZAbCDHjh7zfVNQuf72JSO4+BX5vtRw7YDp7vLHshH/G3u3xXIP/47IYOVzobM
CH+xV/MtoqjzcVvM0sKBCS3j/2ZwM+lgwtao2gvFhjqsWiXLq5PFPgLRO8stvTPh
IpyHsorJNkEt+inlfOxB+RDtJXGv6WmqKNQ1gIKSvEpGRQFPeIOZKLzzLfHSDhYQ
o+P/aAZtwFalwc7743sbanCEGd27PwGdAbgbsD18D2FKWWXpRKQ+RBrCW3Cgqidz
7ZR5fDavowNcC9hViQEKgI2Z6rTWwUR+vCBpAgMBAAGjgaIwgZ8wCQYDVR0TBAIw
ADAdBgNVHQ4EFgQUqKA1ojuEt7JpLYywaoicSc3l+gQwUQYDVR0jBEowSIAUGiqQ
zwN/Xvp+fAyg84mDnOdr2iahGqQYMBYxFDASBgNVBAMMC0Vhc3ktUlNBIENBghQ8
y1aMnI71+Onhc/0xSYfveR4yFjATBgNVHSUEDDAKBggrBgEFBQcDAjALBgNVHQ8E
BAMCB4AwDQYJKoZIhvcNAQELBQADggEBAEykZMheiMK84IIXXqFJcg8Yu+WBKt4j
B0bwLnSKcCOJo/luKU9Eza2quo2xKIQOcwKHalyE2ggGFwPz5MSUc5EJLVCZf2xZ
zuSzk8yA0aPnNhXOGIu0hKF9jB1uXeB3wCHk74XAHsWmyl3r4kyK3LAS7EwEembo
AHinyk0qACplbMdQZzJaj45SDOtb7uoTx53TbfZm6gj16Hx7b/6v/vq10b5LHFw7
2m2QkrIqWoWQM24RObr1uVfGEfj5t+BMDq+gmt/TVDndsKoDNuTnoulkuKKrkpaL
FqZ0S0C/gDoyMnxLua8v3CKKQ6uftTMSl/fL36DhqLBu+Rth8LH5V2w=
-----END CERTIFICATE-----
</cert>

<key>
```

```
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQCqilC9a9bwVhNI
WFc8PemsH1ePTAZpSTNcUId22VjieEIgE3Zny62ZAbCDHjh7zfVNQuf72JSO4+BX
5vtRw7YDp7vLHshH/G3u3xXIP/47IYOVzobMCH+xV/MtoqjzcVvM0sKBCS3j/2Zw
M+lgwtao2gvFhjqsWiXLq5PFPgLRO8stvTPhIpyHsorJNkEt+inlfOxB+RDtJXGv
6WmqKNQ1gIKSvEpGRQFPeIOZKLzzLfHSDhYQo+P/aAZtwFalwc7743sbanCEGd27
PwGdAbgbsD18D2FKWWXpRKQ+RBrCW3Cgqidz7ZR5fDavowNcC9hViQEKgI2Z6rTW
wUR+vCBpAgMBAAECggEAAzkgP8Y+HpZ34JQH2QwAArhobJ1GmT6QjY/5kXPX59bX
RBkuEZXiQuwq4H2FmoRo3JQVQ5ejLiStKKglS5Uv91d4F4WOSjsBfmRhumJIPSib
3qzTtB88f5bukggbGpfDwQYiNjNc1kxtTJVUlbc0/KW7V3k7GmcNRBFK9o0+myL2
4HhAP5dImp/Bd5CSoTyrfCc94gVLDL3RazITGBKOLIvB+HyDR6op5wqNi9zlKewD
BsBh9ncuSX29wsEtMtPYNsIJEaAtMtE9LNQB4ChA33N7DDClpToYh25oc5SofEjp
RFy7tj7mCGVBZQPqA6eDmK0nu6wkJVJ7yVegcIOpcQKBgQDffT97BVu/G+Q6YWsY
HDuFyOYcAL3DgvWd7MT+AgxlZnEUTC+S/i0NkwH43BuCsdV4KiwUSbkw1I4fjGXs
4lhHXJFi/tCtxD4ZEDE5gaj+FCSpHVkZEH0PrN18C7SDTDq1G0htWBlk9sXlciKx
JlPUMEVLuY+u6enq7pTtS850EQKBgQDDWT7R+iBefIza3fSCDxLKsbbQ0hQYkymr
0YIJ6v47erkI5l53vgRAecq4bq4Y42YTPiH7ZrqYKPWYwXxRh3dD9wFRWtac5vxZ
DRYO9t3ZRgTZjAMs8cG5uKgurzfrYrih1gnzKC5YHT/1XchyVaUoCvbKQJz+uwil
XESrs8Xe2QKBgQCQqE1Sn6FHDhKoBy5+qKeObP8k2QX7auT4l6zajRDhAXHoJXgV
uRgIVUNNhYr5CYqXARUO/Lk19h8YJJRExC1H447nePAxyXg4WNbD1j1AWGFyZWCq
bJXGc6nZBOqOoeTOHR7AR+oIBAMMBNiXuatBCQ2RekaafSW/vzX+crbrQQKBgQCC
NSNVTHZ05GnynM0UP73T0z6607xHYRf+iYg22iaZOMTK9Np0z/f/je6cnlF3D2OD
Yf++lYu8Tljdd+JIaZYKfEKpmnXAYYpERR+128ClyTEEXHnlZEMvarDXZT1NrSDJ
5mP9aoPxgZWXE+q+ou2R0jULPKQoejxaFDVFiW9TiQKBgChyzGgTBUL3MpFSzfKA
5VPm1fhfVUj8J/4UsoCTablUwQp1povnUOtQXcTn4ETCtDJLqxeRLOvb7PvYb6uw
psYxK5Ldkgc8CBRVCpTkMAwDaON7B9+s+tfi9RQpo02F4O+eQ4GbSbk5NoB9/cGG
1Y3tcNLbyQVxjj4xcg4+oQW5
-----END PRIVATE KEY-----
</key>

<ca>
-----BEGIN CERTIFICATE-----
MIIDSzCCAjOgAwIBAgIUPMtWjJyO9fjp4XP9MUmH73keMhYwDQYJKoZIhvcNAQEL
BQAwFjEUMBIGA1UEAwwLRWFzeS1SU0EgQ0EwHhcNMjUxMjEwMTgwMzI3WhcNMzUx
```

```
MjA4MTgwMzI3WjAWMRQwEgYDVQQDDAtFYXN5LVJTQSBDQTCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBANG3LlUYzD6jm8cAZmxDfTdQWrvlKLJHM3XiHXQd
3KUh3DkzFfJj4ErIDz/vacd9H64souu6gRhK8V2Gyz0/Z52q/zKdSmCwEgc6O1Ut
ijhpoI7oAiECNUPI72KNGCUbU2dygavQgE9g8+fLzq3unLGvKEtE5pvMOWPfORbE
qW/mOUOZ7rHIF0hpNsPecy489afBxr85TstVU0A+52E3OAe0cGM25mGrUHYdOOH6
LOWdhuo0ue8lUs3dX3CfyA7+ZhoKYqo51uRBzAvw+s8FQ+Rq+dDnlkdMsJtri9qf
Gmgu7KbVzQzIN4tDjGogBb4l/yiNCeVOia0mUjXSgqEwY1kCAwEAAaOBkDCBjTAM
BgNVHRMEBTADAQH/MB0GA1UdDgQWBBQaKpDPA39e+n58DKDziYOc52vaJjBRBgNV
HSMESjBIgBQaKpDPA39e+n58DKDziYOc52vaJqEapBgwFjEUMBIGA1UEAwwLRWFz
eS1SU0EgQ0GCFDzLVoycjvX46eFz/TFJh+95HjIWMAsGA1UdDwQEAwIBBjANBgkq
hkiG9w0BAQsFAAOCAQEAAuyJI1/QgqG/iC/qY9L7kwCiCIOLawgOxrI8dccQNF+h
/SW/hKoRKXsa7D36rcf2ILYU32k8HLPJERys7mF0lEJjtVyvX5F2GCLhP6PZncaq
2WoyktVLaa3ZenO4feRh8xhs87c8KvShcnFXz1TrBx+xY7cXOPlB+5NkCnaOMXXU
M7PUgO0zJdyb+H5++6dMGc32gvhOPOQwD3+sTCUVCRFTPRgW+c2K6+BvnQYkAsyp
eA+AeHvjrXLNsJII652JuIvGpX78wHJqu/CIp5HbvWcvW5qAOZ40/5GhCq8b+GX9
XQEQvMXR7A9gNYPJ/fl/4GZdxgEOp0gZnLqEULGuuA==
-----END CERTIFICATE-----
</ca>

<tls-crypt>
-----BEGIN OpenVPN Static key V1-----
4df196f72a3288a80cb9b84dcc80f072
149c11f4537191279b96eb34d901d9a4
0622d586b665714a7e69633a14cac89f
fe4d45cead09bc5cada13b0c29f89a6e
1e906dc6734b5839fd8be2592ec430b7
f522d46e44a4fba52ca4df74c9527372
ebc7385593d30a8e1d9879fc7f9a970e
a45ad5d33f1071a1b0c31d392ce69867
949e1eafed0169fa7fb0bef2327f847c
48de18a6af53c531e6004303ab13c061
4267ae93be907315c183d780a882e5f9
6c8641fe6652b42aedf670d03a502712
e5ee260509e3d6c376221b33d88b4988
```

```
36a25037ca232c77e6667b8ad94ad2f9
c01e7ef3eaa2e25b7309484a15f11e72
bd667d936568eff0b2985c4653433807
-----END OpenVPN Static key V1-----
</tls-crypt>
```

Voici l'explication détaillée de chaque directive :

Configuration et Mode Client

client

- **Rôle** : Indique qu'il s'agit d'un fichier de configuration pour un client OpenVPN, et non pour un serveur.
- **Implication** : Le client initiera la connexion vers le serveur.

dev tun

- **Rôle** : Spécifie l'utilisation d'un périphérique réseau virtuel de type TUN (Tunnel).
- **Détail** : Le périphérique TUN gère le routage des paquets IP (couche 3 du modèle OSI), créant un tunnel acheminé.

proto udp

- **Rôle** : Définit le protocole de transport à utiliser, qui est UDP (User Datagram Protocol).
- **Détail** : UDP est généralement préféré à TCP pour le VPN car il est plus rapide (il n'y a pas de surcharge pour la correction d'erreurs et la retransmission) et permet d'éviter le “TCP-over-TCP” (problèmes de performance et de latence).

remote 10.0.2.46 1194

- **Rôle** : Indique l'adresse et le port du serveur OpenVPN auquel le client doit se connecter.

- **Détail** :
  - **10.0.2.46** : L'adresse IP (ou le nom d'hôte) du serveur distant.
  - **1194** : Le numéro de port UDP standard sur lequel le serveur OpenVPN est censé écouter.

Stabilité et Sécurité

resolv-retry infinite

- **Rôle** : Permet au client de continuer d'essayer de résoudre l'adresse IP du serveur OpenVPN indéfiniment, en cas d'échec initial (par exemple, si la connexion réseau n'est pas encore établie au démarrage).
- **Utilité** : Très utile pour les clients mobiles ou les ordinateurs portables.

nobind

- **Rôle** : Indique au client de ne pas se lier à un port local spécifique.
- **Détail** : Le client utilisera un port source éphémère (aléatoire), ce qui est le comportement par défaut pour la plupart des clients réseau.

persist-key

- **Rôle** : Empêche le rechargement des clés après une reconnexion ou un redémarrage.
- **Utilité** : Permet de maintenir la connexion et d'accélérer les reconnexions sans avoir à régénérer les clés TLS.

persist-tun

- **Rôle** : Empêche la fermeture et la réouverture de l'interface TUN/TAP virtuelle lors d'un redémarrage ou d'une reconnexion.
- **Utilité** : Cela aide à conserver les routes associées à l'interface VPN, rendant la reconnexion plus transparente.

remote-cert-tls server

- **Rôle** : Ajoute une couche de sécurité en exigeant que le certificat présenté par le serveur inclue la vérification que l'identité est celle d'un serveur.
- **Détail** : Cette directive vérifie le champ key usage et/ou le champ extended key usage du certificat TLS du serveur pour s'assurer que l'homologue distant est bien un serveur OpenVPN et pas un autre client potentiellement malveillant.

auth SHA512

- **Rôle** : Définit l'algorithme de hachage HMAC (Hashed Message Authentication Code) utilisé pour l'authentification des paquets du canal de données.
- **Détail** : Le client utilisera SHA512 au lieu du SHA1 par défaut pour l'authentification des messages (la vérification que le paquet n'a pas été altéré en transit).

ignore-unknown-option block-outside-dns

- **Rôle** : Indique à OpenVPN d'ignorer une option de configuration qui n'est pas prise en charge ou n'est pas reconnue par cette version du client.
- **Détail** : L'option block-outside-dns est spécifique à certaines plateformes (comme OpenVPN pour Windows) pour empêcher les fuites DNS en dehors du tunnel. En utilisant ignore-unknown-option, la configuration reste compatible avec des clients qui ne supportent pas cette fonctionnalité sans provoquer d'erreur fatale.

Journalisation

verb 3

- **Rôle** : Définit le niveau de verbosité (détail) du journal (log) d'OpenVPN.
- **Détail** : Un niveau de 3 est généralement suffisant pour une utilisation quotidienne, fournissant les avertissements mineurs et les informations de connexion, mais sans surcharger le journal. (Le niveau maximum est 15, le niveau 0 est minimal).

4.4 - Les Clefs du Client

1. Le Certificat Client (<cert>)

Ce bloc contient le certificat public X.509 du client. Il prouve l'identité du client et est utilisé par le serveur pour l'authentification.

Détails Clés du Certificat :

Subject: CN=client

- **Signification** : C'est l'identité de l'entité que ce certificat représente. Dans ce cas, il s'agit du client OpenVPN, identifié par le nom commun (Common Name) client.

Issuer: CN=Easy-RSA CA

- **Signification** : C'est l'entité qui a émis et signé ce certificat, l'Autorité de Certification (CA) nommée Easy-RSA CA. C'est l'élément de confiance fondamental.

Validity (Not Before / Not After)

- **Signification** : Définit la période de validité du certificat (du 10 décembre 2025 au 8 décembre 2035).

Signature Algorithm: sha256WithRSAEncryption

- **Signification** : Le certificat a été signé par la CA en utilisant l'algorithme de hachage SHA-256 combiné à la cryptographie RSA.

Subject Public Key Info: Public-Key: (2048 bit)

- **Signification** : Contient la clé publique du client (2048 bits RSA). Cette clé est publique et est utilisée par le serveur pour établir la session TLS cryptée avec le client.

X509v3 Extended Key Usage: TLS Web Client Authentication

- **Signification** : Cruciale pour OpenVPN. Cette extension indique que le certificat est destiné uniquement à l'authentification d'un client dans une connexion TLS. Elle empêche l'utilisation de ce certificat pour d'autres usages (comme un certificat serveur) et est souvent vérifiée par la directive remote-cert-tls server sur la configuration du serveur.

X509v3 Basic Constraints: CA:FALSE

- **Signification** : Indique que ce certificat n'est pas une Autorité de Certification et ne peut pas signer d'autres certificats.

2. La Clé Privée (<key>)

Ce bloc contient la clé privée RSA du client.

Rôle et Importance

- **Rôle** : La clé privée est le secret absolu du client. Elle est mathématiquement liée à la clé publique présente dans le certificat.
- **Fonction** : Le client utilise cette clé privée pour :
  - Déchiffrer les données chiffrées par le serveur lors de l'établissement de la connexion TLS.
  - Prouver qu'il est bien le propriétaire du certificat lors de l'établissement de la liaison sécurisée (le “handshake”).
  - Sécurité : Cette clé ne doit jamais être partagée. Si elle est compromise, n'importe qui peut se faire passer pour ce client auprès du serveur VPN.

3. Le Certificat de l'Autorité de Certification (<ca>)

Ce bloc contient le certificat public de l'Autorité de Certification (CA).

- **Rôle et Importance**
  - Rôle Fondamental : C'est la racine de confiance de tout le système OpenVPN.

Subject: CN=Easy-RSA CA et Issuer: CN=Easy-RSA CA

- **Signification** : Le fait que l'émetteur et le sujet soient identiques indique qu'il s'agit d'un certificat auto-signé (Root CA).
- **Fonction** : Le client utilise cette clé pour vérifier que le certificat du serveur (qui n'est pas inclus ici) a bien été signé par cette CA. Sans cette vérification, le client ne peut pas faire confiance au serveur, ce qui empêche les attaques de type “homme du milieu” (Man-in-the-Middle).

X509v3 Basic Constraints: CA:TRUE

- **Signification** : Confirme que ce certificat est bien destiné à être une Autorité de Certification capable de signer d'autres certificats (clients et serveurs).

4. La Clé Statique TLS-Crypt (<tls-crypt>)

Ce bloc contient une clé secrète symétrique partagée (Statique Key) utilisée pour renforcer la sécurité du canal de contrôle OpenVPN.

- **Rôle et Avantages**
  - Rôle : C'est une clé secrète pré-partagée que le client et le serveur OpenVPN utilisent avant même de démarrer la liaison TLS.
- **Fonction**
  - Obfuscation (Brouillage) : Elle chiffre la poignée de main TLS initiale. Cela rend le trafic OpenVPN méconnaissable aux tentatives de détection externes (Deep Packet Inspection), y compris l'identification des paquets de contrôle TLS.
  - Protection DoS : Les paquets non signés avec cette clé sont immédiatement abandonnés par le serveur, ce qui le protège contre les scans de ports non-OpenVPN ou les attaques par déni de service distribué (DDoS).
- **Détail** : L'en-tête —–BEGIN OpenVPN Static key V1—– indique que c'est une clé statique pré-partagée. OpenVPN utilise cette clé pour le chiffrement des métadonnées du canal de contrôle.
- **Résumé du Flux de Confiance (PKI)**
  - Authentification Serveur : Le client utilise le certificat <ca> pour vérifier la signature sur le certificat du serveur.
  - Authentification Client : Le serveur utilise son propre certificat <ca> pour vérifier la signature sur le certificat <cert> du client.
  - Établissement du Tunnel : La clé privée **key** et la clé publique du serveur (dans son certificat) sont utilisées pour établir une clé de session

symétrique.
- Sécurité Renforcée : La clé symétrique **tls-crypt** est utilisée pour masquer les métadonnées de cette poignée de main TLS.

4.5 - Tester la Configuration

Retournez dans la VM **CentOS8_10.0.2.45_SSH** et démarrez la connexion client avec le fichier **client.ovpn** :

```
[root@centos8 ~]# openvpn --config client.ovpn
Wed Dec 10 13:15:06 2025 OpenVPN 2.4.12 x86_64-redhat-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11]
[MH/PKTINFO] [AEAD] built on Nov 10 2023
Wed Dec 10 13:15:06 2025 library versions: OpenSSL 1.1.1k  FIPS 25 Mar 2021, LZO 2.08
Wed Dec 10 13:15:06 2025 Outgoing Control Channel Encryption: Cipher 'AES-256-CTR' initialized with 256 bit key
Wed Dec 10 13:15:06 2025 Outgoing Control Channel Encryption: Using 256 bit message hash 'SHA256' for HMAC
authentication
Wed Dec 10 13:15:06 2025 Incoming Control Channel Encryption: Cipher 'AES-256-CTR' initialized with 256 bit key
Wed Dec 10 13:15:06 2025 Incoming Control Channel Encryption: Using 256 bit message hash 'SHA256' for HMAC
authentication
Wed Dec 10 13:15:06 2025 TCP/UDP: Preserving recently used remote address: [AF_INET]10.0.2.46:1194
Wed Dec 10 13:15:06 2025 Socket Buffers: R=[212992->212992] S=[212992->212992]
Wed Dec 10 13:15:06 2025 UDP link local: (not bound)
Wed Dec 10 13:15:06 2025 UDP link remote: [AF_INET]10.0.2.46:1194
Wed Dec 10 13:15:06 2025 TLS: Initial packet from [AF_INET]10.0.2.46:1194, sid=1b107770 8c7e6805
Wed Dec 10 13:15:06 2025 VERIFY OK: depth=1, CN=Easy-RSA CA
Wed Dec 10 13:15:06 2025 VERIFY KU OK
Wed Dec 10 13:15:06 2025 Validating certificate extended key usage
Wed Dec 10 13:15:06 2025 ++ Certificate has EKU (str) TLS Web Server Authentication, expects TLS Web Server
Authentication
Wed Dec 10 13:15:06 2025 VERIFY EKU OK
Wed Dec 10 13:15:06 2025 VERIFY OK: depth=0, CN=server
Wed Dec 10 13:15:06 2025 WARNING: 'cipher' is present in local config but missing in remote config, local='cipher
BF-CBC'
Wed Dec 10 13:15:06 2025 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA384, 2048 bit RSA
Wed Dec 10 13:15:06 2025 [server] Peer Connection Initiated with [AF_INET]10.0.2.46:1194
```

```
Wed Dec 10 13:15:07 2025 SENT CONTROL [server]: 'PUSH_REQUEST' (status=1)
Wed Dec 10 13:15:07 2025 PUSH: Received control message: 'PUSH_REPLY,redirect-gateway def1 bypass-dhcp,dhcp-
option DNS 8.8.8.8,dhcp-option DNS 8.8.4.4,block-outside-dns,route-gateway 10.8.0.1,topology subnet,ping 10,ping-
restart 120,ifconfig 10.8.0.2 255.255.255.0,peer-id 0,cipher AES-256-GCM'
Wed Dec 10 13:15:07 2025 Unrecognized option or missing or extra parameter(s) in [PUSH-OPTIONS]:4: block-outside-
dns (2.4.12)
Wed Dec 10 13:15:07 2025 OPTIONS IMPORT: timers and/or timeouts modified
Wed Dec 10 13:15:07 2025 OPTIONS IMPORT: --ifconfig/up options modified
Wed Dec 10 13:15:07 2025 OPTIONS IMPORT: route options modified
Wed Dec 10 13:15:07 2025 OPTIONS IMPORT: route-related options modified
Wed Dec 10 13:15:07 2025 OPTIONS IMPORT: --ip-win32 and/or --dhcp-option options modified
Wed Dec 10 13:15:07 2025 OPTIONS IMPORT: peer-id set
Wed Dec 10 13:15:07 2025 OPTIONS IMPORT: adjusting link_mtu to 1624
Wed Dec 10 13:15:07 2025 OPTIONS IMPORT: data channel crypto options modified
Wed Dec 10 13:15:07 2025 Data Channel: using negotiated cipher 'AES-256-GCM'
Wed Dec 10 13:15:07 2025 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 256 bit key
Wed Dec 10 13:15:07 2025 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 256 bit key
Wed Dec 10 13:15:07 2025 ROUTE_GATEWAY 10.0.2.1/255.255.255.0 IFACE=ens18 HWADDR=4e:b1:31:bd:5d:b2
Wed Dec 10 13:15:07 2025 TUN/TAP device tun0 opened
Wed Dec 10 13:15:07 2025 TUN/TAP TX queue length set to 100
Wed Dec 10 13:15:07 2025 /sbin/ip link set dev tun0 up mtu 1500
Wed Dec 10 13:15:07 2025 /sbin/ip addr add dev tun0 10.8.0.2/24 broadcast 10.8.0.255
Wed Dec 10 13:15:07 2025 /sbin/ip route add 10.0.2.46/32 dev ens18
Wed Dec 10 13:15:07 2025 /sbin/ip route add 0.0.0.0/1 via 10.8.0.1
Wed Dec 10 13:15:07 2025 /sbin/ip route add 128.0.0.0/1 via 10.8.0.1
Wed Dec 10 13:15:07 2025 WARNING: this configuration may cache passwords in memory -- use the auth-nocache option
to prevent this
Wed Dec 10 13:15:07 2025 Initialization Sequence Completed
```

Vérifiez la présence de l'interface **tun0** :

```
[root@centos8 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

```
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 4e:b1:31:bd:5d:b2 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.45/24 brd 10.0.2.255 scope global noprefixroute ens18
       valid_lft forever preferred_lft forever
    inet6 fe80::86b6:8d39:cab2:d84d/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: ens19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether ea:c8:86:9e:73:a6 brd ff:ff:ff:ff:ff:ff
4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:79:02:66 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
5: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel master virbr0 state DOWN group default qlen 1000
    link/ether 52:54:00:79:02:66 brd ff:ff:ff:ff:ff:ff
6: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 100
    link/none
    inet 10.8.0.2/24 brd 10.8.0.255 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::f71b:f026:9ffc:dfa7/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
```

Vérifiez que vous pouvez contacter le serveur OpenVPN :

```
[root@centos8 ~]# ping 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=0.993 ms
64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=0.780 ms
64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=0.824 ms
64 bytes from 10.8.0.1: icmp_seq=4 ttl=64 time=0.771 ms
^C
```

```
--- 10.8.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3084ms
rtt min/avg/max/mdev = 0.771/0.842/0.993/0.089 ms
```

Retournez dans la VM **Debian12_10.0.2.46_SSH** et vérifiez la présence de l'interface **tun0** :

```
root@debian12:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 56:a3:fd:18:02:6d brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    inet 10.0.2.46/24 brd 10.0.2.255 scope global noprefixroute ens18
       valid_lft forever preferred_lft forever
    inet6 fe80::54a3:fdff:fe18:26d/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 500
    link/none
    inet 10.8.0.1/24 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::c401:deef:3b77:6193/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
```

Dernièrement, vérifiez la communication avec le client :

```
root@debian12:~# ping 10.8.0.2
PING 10.8.0.2 (10.8.0.2) 56(84) bytes of data.
64 bytes from 10.8.0.2: icmp_seq=1 ttl=64 time=0.686 ms
64 bytes from 10.8.0.2: icmp_seq=2 ttl=64 time=0.832 ms
64 bytes from 10.8.0.2: icmp_seq=3 ttl=64 time=0.776 ms
```

```
^C
--- 10.8.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2034ms
rtt min/avg/max/mdev = 0.686/0.764/0.832/0.060 ms
```

LAB #5 - Mise en place d'un serveur Wireguard

5.1 - Installation et Configuration du Serveur

Commencez par installer **wireguard** dans la VM **Wireguard_10.0.2.47_SSH** :

```
trainee@wireguard:~$ su -
Password: fenestros

root@wireguard:~# apt install wireguard

root@wireguard:~# exit
```

Restez dans le serveur **wireguard** et générez la clef privée et la clef publique :

```
trainee@wireguard:~$ wg genkey | tee privatekey | wg pubkey > publickey

trainee@wireguard:~$ ls
Desktop    Downloads  Pictures    Public     Templates
Documents  Music      privatekey  publickey  Videos

trainee@wireguard:~$ cat privatekey
qN/H49BMnAiXunK0F694NAFsJ/fpjWKjdZ7U+Ew1DXQ=
```

Copiez la clef privée et créez le fichier /etc/wireguard/wg0.conf :

```
trainee@wireguard:~$ su -
Password: fenestros

root@wireguard:~# vi /etc/wireguard/wg0.conf

root@wireguard:~# cat /etc/wireguard/wg0.conf
[Interface]
PrivateKey=qN/H49BMnAiXunK0F694NAFsJ/fpjWKjdZ7U+Ew1DXQ=
Address=10.0.0.1/8
SaveConfig=true
#PostUp=iptables -A forward -i wg0 -j ACCEPT; iptables -t nat -A POSTROUTING -o ens18 -j MAQUERADE
#PostDown=iptables -D forward -i wg0 -j ACCEPT; iptables -t nat -D POSTROUTING -o ens18 -j MAQUERADE
ListenPort=51820
```

Démarrez votre interface VPN :

```
root@wireguard:~# wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 10.0.0.1/8 dev wg0
[#] ip link set mtu 1420 up dev wg0
```

Consultez la clef public du serveur Wireguard grâce à la commande **wg** :

```
root@wireguard:~# wg
interface: wg0
  public key: oX9NnLPEOz6+bsVtARBy79WnbvrcU0sQ54rqFEZj5XU=
  private key: (hidden)
  listening port: 51280
```

Vérifiez que voyez bien l'interface **wg0** :

```
root@wireguard:~# ip link
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether bc:24:11:d4:dc:cc brd ff:ff:ff:ff:ff:ff
    altname enp0s18
4: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1420 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/none
```

5.2 - Installation et Configuration du Client

Commencez par installer **wireguard** dans la VM **Debian12_10.0.2.46_SSH** :

```
trainee@debian12:~$ su -
Password: fenestros

root@debian12:~# apt install wireguard

root@debian12:~# exit
```

Créez une clef secrète et une clef publique :

```
trainee@debian12:~$ wg genkey | tee privatekey | wg pubkey > publickey

trainee@debian12:~$ cat privatekey
IOQnfB/Yip4X5LkCnXgUtTxPxRSu31ZgFsTIXfwVVVE=
```

Copiez la valeur de la clef privée et créez le fichier **/etc/wireguard/wg0.conf** :

```
trainee@debian12:~$ su -
Password: fenestros

root@debian12:~# vi /etc/wireguard/wg0.conf
```

```
root@debian12:~# cat /etc/wireguard/wg0.conf
[Interface]
PrivateKey=IOQnfB/Yip4X5LkCnXgUtTxPxRSu31ZgFsTIXfwVVVE=
Address=10.0.0.2/8
SaveConfig=true

[Peer]
PublicKey=oX9NnLPEOz6+bsVtARBy79WnbvrcU0sQ54rqFEZj5XU=
Endpoint=10.0.2.47:51820
AllowedIPs=0.0.0.0/0
```

Important - La valeur **0.0.0.0/0** de la directive **AllowedIPs** forcera tout le trafic du client à passer par le VPN.

Démarrez l'interface **wg0** :

```
root@debian12:~# wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 10.0.0.2/8 dev wg0
[#] ip link set mtu 1420 up dev wg0
[#] wg set wg0 fwmark 51820
[#] ip -4 route add 0.0.0.0/0 dev wg0 table 51820
[#] ip -4 rule add not fwmark 51820 table 51820
[#] ip -4 rule add table main suppress_prefixlength 0
[#] sysctl -q net.ipv4.conf.all.src_valid_mark=1
[#] nft -f /dev/fd/63
```

Visualisez et copiez la valeur de la clef publique du client :

```
root@debian12:~# wg
```

```
interface: wg0
  public key: 3PVLRBC0N2g42KOdPFe0HgYZNpB38PgROAavWeZ0/hw=
  private key: (hidden)
  listening port: 42207
  fwmark: 0xca6c

peer: oX9NnLPEOz6+bsVtARBy79WnbvrcU0sQ54rqFEZj5XU=
  endpoint: 10.0.2.47:51820
  allowed ips: 0.0.0.0/0
  transfer: 0 B received, 2.17 KiB sent
```

Retournez au serveur wireguard et informez-le que le client est autorisé à s'y connecter :

```
root@wireguard:~# wg set wg0 peer 3PVLRBC0N2g42KOdPFe0HgYZNpB38PgROAavWeZ0/hw= allowed-ips 10.0.0.2/32
```

Retournez au client et arrêtez l'interface **wg0** :

```
root@debian12:~# wg-quick down wg0
[#] wg showconf wg0
[#] ip -4 rule delete table 51820
[#] ip -4 rule delete table main suppress_prefixlength 0
[#] ip link delete dev wg0
[#] nft -f /dev/fd/63
```

Ajoutez **PersistentKeepalive=30** au fichier **/etc/wireguard/wg0.conf** :

```
root@debian12:~# vi /etc/wireguard/wg0.conf

root@debian12:~# cat /etc/wireguard/wg0.conf
[Interface]
Address = 10.0.0.2/8
SaveConfig = true
ListenPort = 42207
FwMark = 0xca6c
```

```
PrivateKey = IOQnfB/Yip4X5LkCnXgUtTxPxRSu31ZgFsTIXfwVVVE=

[Peer]
PublicKey = oX9NnLPEOz6+bsVtARBy79WnbvrcU0sQ54rqFEZj5XU=
AllowedIPs = 0.0.0.0/0
Endpoint = 10.0.2.47:51820
PersistentKeepalive=30
```

Redémarrez l'interface **wg0** :

```
root@debian12:~# wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 10.0.0.2/8 dev wg0
[#] ip link set mtu 1420 up dev wg0
[#] ip -4 route add 0.0.0.0/0 dev wg0 table 51820
[#] ip -4 rule add not fwmark 51820 table 51820
[#] ip -4 rule add table main suppress_prefixlength 0
[#] sysctl -q net.ipv4.conf.all.src_valid_mark=1
[#] nft -f /dev/fd/63
```

Constatez la mise ne place de votre modification :

```
root@debian12:~# wg
interface: wg0
  public key: 3PVLRBC0N2g42KOdPFe0HgYZNpB38PgROAavWeZ0/hw=
  private key: (hidden)
  listening port: 42207
  fwmark: 0xca6c

peer: oX9NnLPEOz6+bsVtARBy79WnbvrcU0sQ54rqFEZj5XU=
  endpoint: 10.0.2.47:51820
  allowed ips: 0.0.0.0/0
  transfer: 0 B received, 296 B sent
```

```
  persistent keepalive: every 30 seconds
```

5.3 - Tester la Configuration

```
root@debian12:~# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.685 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.572 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.776 ms
^C
--- 10.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2028ms
rtt min/avg/max/mdev = 0.572/0.677/0.776/0.083 ms
```

```
root@wireguard:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.607 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.649 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.764 ms
^C
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2044ms
rtt min/avg/max/mdev = 0.607/0.673/0.764/0.066 ms
```
