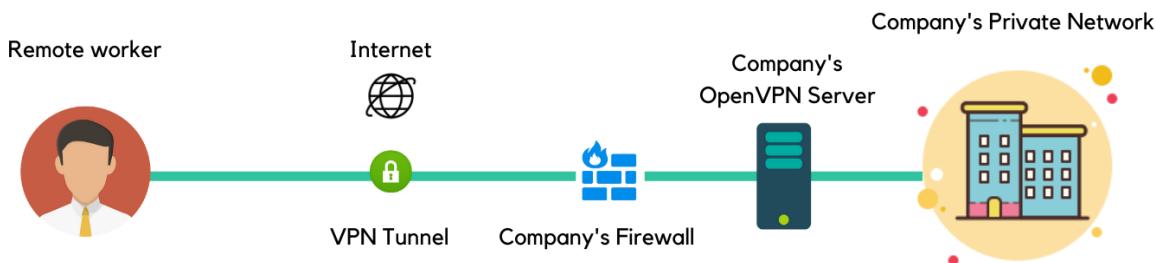


# TP : Installation et Configuration d'un VPN avec OpenVPN (Client/Serveur)



- Réalisé par : Chaimae Bouassab
- Professeur : Mr. GHADI Abderrahim
- Filière : Master Sécurité IT & Big Data
- Année universitaire : 2024-2025

## 1. Introduction

Dans ce TP, l'objectif principal est de mettre en place un réseau privé virtuel (VPN) sécurisé entre un serveur Linux (Kali) et un client Windows, en utilisant OpenVPN.

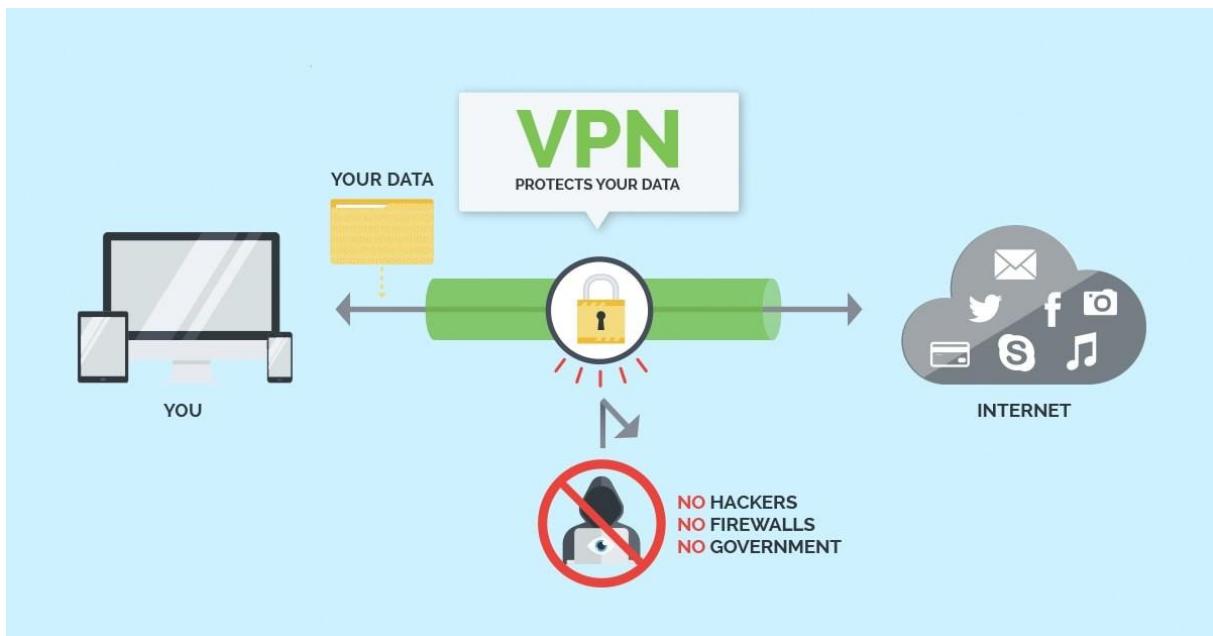
OpenVPN est une solution open source populaire qui permet de créer des **tunnels VPN chiffrés** pour sécuriser la communication sur des réseaux publics.

Ce projet implique la génération de certificats SSL/TLS, la configuration du serveur VPN et du client, ainsi que la vérification de la connectivité et du chiffrement.

Les principaux outils utilisés sont Kali Linux (serveur), Windows (client) et OpenVPN avec son interface graphique sous Windows (OpenVPN GUI).

## 2. Préparation de l'environnement serveur

Pour commencer, nous avons configuré le serveur VPN sous Kali Linux. Cette étape est essentielle car elle établit la base sécurisée du réseau VPN.



## Installation des outils nécessaires

Nous avons tout d'abord installé les paquets essentiels pour OpenVPN et la gestion des certificats avec la commande suivante :

```
chaimae@kali: /usr/share/easy-rsa
.....+....+.....+.....+.....+....+
+...+...+...+....+...+.....+....+....+...
+....+...+.....+...+...+....+....+....+....+...
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Common Name (eg: your user, host, or server name) [Easy-RSA CA]:
Notice
-----
CA creation complete. Your new CA certificate is at:
* /usr/share/easy-rsa/pki/ca.crt

Create an OpenVPN TLS-AUTH|TLS-CRYPT-V1 key now: See 'help gen-tls'

Build-ca completed successfully.

└─(chaimae㉿kali)-[~/usr/share/easy-rsa]
└─$
```

openvpn permet de lancer le service VPN tandis que easy-rsa facilite la génération et la gestion des certificats SSL/TLS indispensables à l'authentification et au chiffrement.

## Initialisation du PKI (Public Key Infrastructure)

Ensuite, nous avons initialisé l'infrastructure de clés publiques (PKI) avec Easy-RSA, qui va nous permettre de générer l'autorité de certification (CA) et les certificats pour le serveur et les clients.

Lors de la génération de la CA (build-ca), un mot de passe est demandé pour protéger la clé privée de l'autorité.

## Génération des certificats et clés

Puis, nous avons généré la paire clé-certificat du serveur, des clients, ainsi que les fichiers nécessaires pour le chiffrement TLS :



```
Private-Key and Public-Certificate-Request files created.  
Your files are:  
* req: /usr/share/easy-rsa/pki/reqs/server1.req  
* key: /usr/share/easy-rsa/pki/private/server1.key
```

You are about to sign the following certificate:

```
Requested CN:      'server1'  
Requested type:   'server'  
Valid for:        '825' days
```

```
subject= commonName = server1
```

Type the word 'yes' to continue, or any other input to abort.  
Confirm requested details: yes

```
private-Key and Public-Certificate-Request files created.  
Your files are:  
: req: /usr/share/easy-rsa/pki/reqs/client1.req  
: key: /usr/share/easy-rsa/pki/private/client1.key  
  
You are about to sign the following certificate:  
  
Requested CN:      'client1'  
Requested type:    'client'  
Valid for:        '825' days  
  
subject=  
    commonName          = client1  
  
Type the word 'yes' to continue, or any other input to abort.  
Confirm requested details: yes  
  
Using configuration from /usr/share/easy-rsa/pki/0353d998/temp.6.1  
Enter pass phrase for /usr/share/easy-rsa/pki/private/ca.key:
```

```
└─(chainae㉿kali)-[/usr/share/easy-rsa]  
└─$ sudo openvpn --genkey secret ./pki/ta.key
```

- **ta.key : clé TLS authentifiée pour renforcer la sécurité contre les attaques par rejet**
- **Configuration du serveur OpenVPN**
- Nous avons ensuite copié les certificats et clés dans le dossier de configuration d'OpenVPN et configuré le serveur via le fichier **server.conf**.
- Le serveur est configuré pour utiliser le protocole UDP, avec un sous-réseau VPN dédié (par exemple 10.8.0.0/24), et pour activer le chiffrement et l'authentification mutuelle des certificats.

```
(chaimae㉿kali)-[~/usr/share/easy-rsa]
└─$ sudo cp -pR pki/{issued,private,ca.crt,dh.pem,ta.key} /etc/openvpn/server/
└─$ sudo cp /usr/share/doc/openvpn/examples/sample-config-files/server.conf
      /etc/openvpn/server/
```

```
GNU nano 8.3          /etc/openvpn/server/server.conf
# rules for the TUN/TAP interface.
# On non-Windows systems, you can give
# an explicit unit number, such as tun0.
# On Windows, use "dev-node" for this.
# On most systems, the VPN will not function
# unless you partially or fully disable/open
# the firewall for the TUN/TAP interface.
;dev tap
dev tun

# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel if you
# have more than one.
# You may need to selectively disable the
# Windows firewall for the TAP adapter.
# Non-Windows systems usually don't need this.
;dev-node MyTap

# SSL/TLS root certificate (ca), certificate
# (cert), and private key (key). Each client
```

^G Help ^O Write Out ^F Where Is ^K Cut ^T Execute  
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify

GNU nano 8.3 /etc/openvpn/server/add-bridge.sh

```
#!/bin/bash
IF=eth0
VPNIF=tun0

echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -A FORWARD -i ${VPNIF} -j ACCEPT
iptables -t nat -A POSTROUTING -o ${IF} -j MASQUERADE
```

[ Read 7 lines ]

^G Help ^O Write Out ^F Where Is ^K Cut ^T Execute  
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify

```
(chaimae㉿kali)-[/usr/share/easy-rsa]
$ sudo nano /etc/openvpn/server/remove-bridge.sh

(chaimae㉿kali)-[/usr/share/easy-rsa]
$ sudo chmod 700 /etc/openvpn/server/add-bridge.sh /etc/openvpn/server/remove-bridge.sh
```

```
chaimae@kali: /usr/share/easy-rsa
GNU nano 8.3  /lib/systemd/system/openvpn-server@.service *
Documentation=https://openvpn.net/community-resources/reference-manual>
Documentation=https://community.openvpn.net/openvpn/wiki/HOWTO

[Service]
Type=notify
PrivateTmp=true
WorkingDirectory=/etc/openvpn/server
ExecStart=/usr/sbin/openvpn --status %t/openvpn-server/status-%i.log ->
CapabilityBoundingSet=CAP_IPC_LOCK CAP_NET_ADMIN CAP_NET_BIND_SERVICE >
LimitNPROC=10
DeviceAllow=/dev/null rw
DeviceAllow=/dev/net/tun rw
ProtectSystem=true
ProtectHome=true
KillMode=process
RestartSec=5s
Restart=on-failure
ExecStartPost=/etc/openvpn/server/add-bridge.sh
ExecStopPost=/etc/openvpn/server/remove-bridge.sh
```

^G Help ^O Write Out ^F Where Is ^K Cut ^T Execute  
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify

- Lancement du serveur et vérification

Enfin, le serveur OpenVPN a été lancé avec la commande :

```
sudo systemctl start openvpn@server
```

```
(chaimae㉿kali)-[/usr/share/easy-rsa]
$ sudo systemctl enable --now openvpn-server@server

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

(chaimae㉿kali)-[/usr/share/easy-rsa]
$ sudo systemctl status openvpn-server@server
```

- `openvpn-server@server.service` - OpenVPN service for server
  - Loaded: loaded (/usr/lib/systemd/system/openvpn-server@.service;)
  - Active: active (running) since Sat 2025-05-24 10:39:19 EDT; 13s ago
  - Invocation: 237ac63adcd44e609c6e4e4fd790a3f5
  - Docs: man:openvpn(8)  
[https://openvpn.net/community-resources/reference-manual->](https://openvpn.net/community-resources/reference-manual)  
<https://community.openvpn.net/openvpn/wiki/HOWTO>
  - Process: 6101 ExecStartPost=/etc/openvpn/server/add-bridge.sh (cod...)

```
Invocation: 237ac63adcd44e609c6e4e4fd790a3f5
    Docs: man:openvpn(8)
          https://openvpn.net/community-resources/reference-manual->
          https://community.openvpn.net/openvpn/wiki/HOWTO
Process: 6101 ExecStartPost=/etc/openvpn/server/add-bridge.sh (cod>
Main PID: 6100 (openvpn)
    Status: "Initialization Sequence Completed"
    Tasks: 1 (limit: 4812)
   Memory: 2.7M (peak: 3.9M)
      CPU: 422ms
     CGroup: /system.slice/system-openvpn\x2dserver.slice/openvpn-serv>
              └─6100 /usr/sbin/openvpn --status /run/openvpn-server/sta>

May 24 10:39:18 kali openvpn[6100]: net_addr_v4_add: 10.8.0.1/24 dev tun0
May 24 10:39:18 kali openvpn[6100]: Could not determine IPv4/IPv6 prot>
May 24 10:39:18 kali openvpn[6100]: Socket Buffers: R=[212992->212992]>
May 24 10:39:18 kali openvpn[6100]: UDPv4 link local (bound): [AF_INET]10.8.0.1<-->
May 24 10:39:18 kali openvpn[6100]: UDPv4 link remote: [AF_UNSPEC]
May 24 10:39:18 kali openvpn[6100]: MULTI: multi_init called, r=256 v=>
May 24 10:39:18 kali openvpn[6100]: IFCONFIG POOL IPv4: base=10.8.0.2 >
```

Puis, nous avons vérifié le statut du service afin de s'assurer que l'initialisation s'est bien déroulée sans erreur, et que le serveur est prêt à accepter les connexions :

Le serveur VPN fonctionne correctement, il écoute sur le port 1194 et crée une interface virtuelle tun0 avec l'adresse IP 10.8.0.1.

Cette phase est cruciale car elle garantit la création d'un environnement sécurisé. La génération correcte des certificats et la bonne configuration du serveur assurent l'authentification des clients et la confidentialité des données transmises. L'utilisation d'Easy-RSA simplifie la gestion des clés et rend la procédure plus accessible.

### 3. Préparation du client

#### Création du dossier client

Sur la machine cliente, nous avons créé un dossier spécifique pour stocker tous les fichiers nécessaires à la connexion VPN. Ce dossier permet d'organiser les certificats, clés et fichiers de configuration.

```
(chaimae㉿kali)-[~/Desktop]
$ ls ~/Desktop/client_vpn/
ca.crt  client1.crt  client1.key  ta.key
```

#### Transfert des certificats du serveur vers le client

Nous avons copié les certificats et clés générés sur le serveur vers le dossier client. Cela inclut :

- ca.crt (certificat de l'autorité de certification)
- client1.crt (certificat client)
- client1.key (clé privée client)
- ta.key (clé TLS authentifiée)

```
(chaimae㉿kali)-[/usr/share/easy-rsa]
$ sudo cp /usr/share/easy-rsa/pki/ca.crt ~/Desktop/client_vpn/
$ sudo cp /usr/share/easy-rsa/pki/issued/client1.crt ~/Desktop/client_vpn/
$ sudo cp /usr/share/easy-rsa/pki/private/client1.key ~/Desktop/client_vp
n/
$ sudo cp /usr/share/easy-rsa/pki/ta.key ~/Desktop/client_vpn/
```

```
(chaimae㉿kali)-[~/Desktop]
$ ls /mnt/shared/
ca.crt  client1.crt  client1.key  client.ovpn  ta.key
(chaimae㉿kali)-[~/Desktop]
$
```

Le transfert s'est fait via un dossier partagé entre la machine virtuelle Kali et la machine hôte Windows, en utilisant des droits root pour éviter les erreurs de permission.

```
Command Prompt      + | v
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

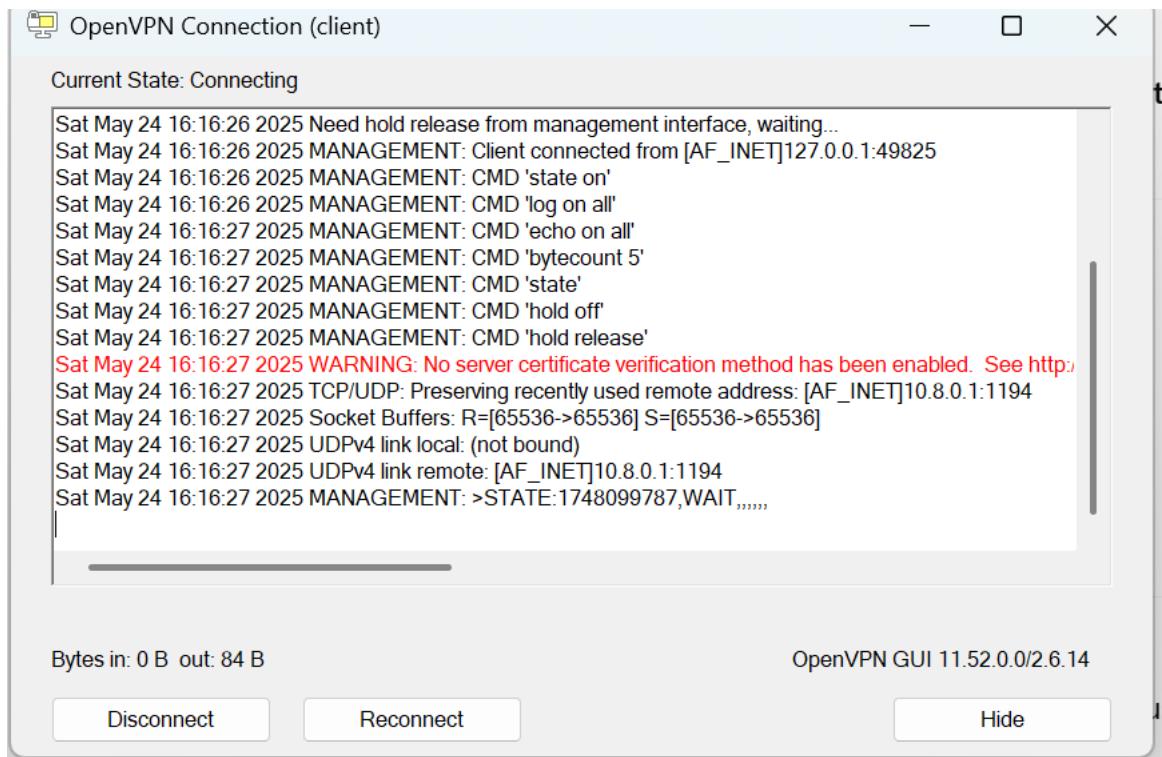
C:\Users\surface>ping 10.8.0.1

Pinging 10.8.0.1 with 32 bytes of data:
Request timed out.
Reply from 81.192.249.77: TTL expired in transit.
Reply from 81.192.249.77: TTL expired in transit.
Reply from 81.192.249.78: TTL expired in transit.

Ping statistics for 10.8.0.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
C:\Users\surface>
```

## Test de connectivité avec ping

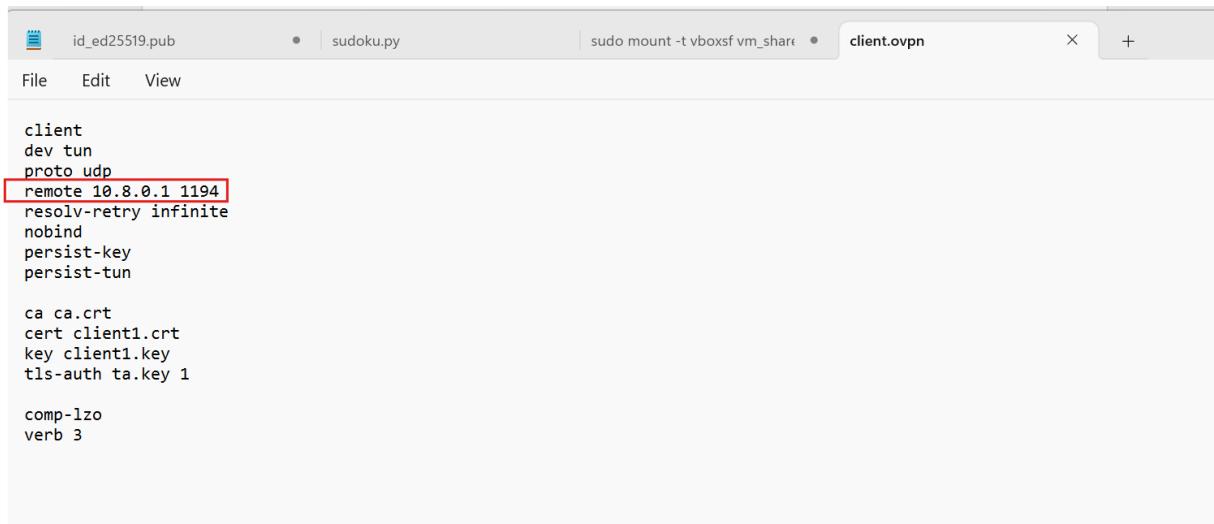
Depuis la machine cliente, nous avons testé la connectivité vers le serveur VPN avec la commande : ping 10.8.0.1



```
(chaimae㉿kali)-[~/usr/share/easy-rsa]
$ ip a | grep inet

    inet 127.0.0.1/8 scope host lo
        inet6 ::1/128 scope host noprefixroute
    inet 172.20.10.2/28 brd 172.20.10.15 scope global dynamic noprefi
oute eth0
        inet6 fe80::a00:27ff:fea5:5b27/64 scope link noprefixroute
    inet 10.8.0.1/24 scope global tun0
        inet6 fe80::2062:617e:f289:360/64 scope link stable-privacy proto
kernel_ll

(chimae㉿kali)-[~/usr/share/easy-rsa]
```



```
id_ed25519.pub      sudoku.py      sudo mount -t vboxsf vm_share  client.ovpn
File   Edit   View

client
dev tun
proto udp
remote 10.8.0.1 1194
resolv-retry infinite
nobind
persist-key
persist-tun

ca ca.crt
cert client1.crt
key client1.key
tls-auth ta.key 1

comp-lzo
verb 3
```

**10.8.0.1 est l'IP interne du tunnel VPN, pas l'IP du serveur à utiliser pour se connecter.**

```

GNU nano 8.3          chaimae@kali:~/Desktop          /mnt/shared/client.ovpn
client
dev tun
proto udp
remote 172.20.10.2 1194
resolv-retry infinite
nobind
persist-key
persist-tun

ca ca.crt
cert client1.crt
key client1.key
tls-auth ta.key 1

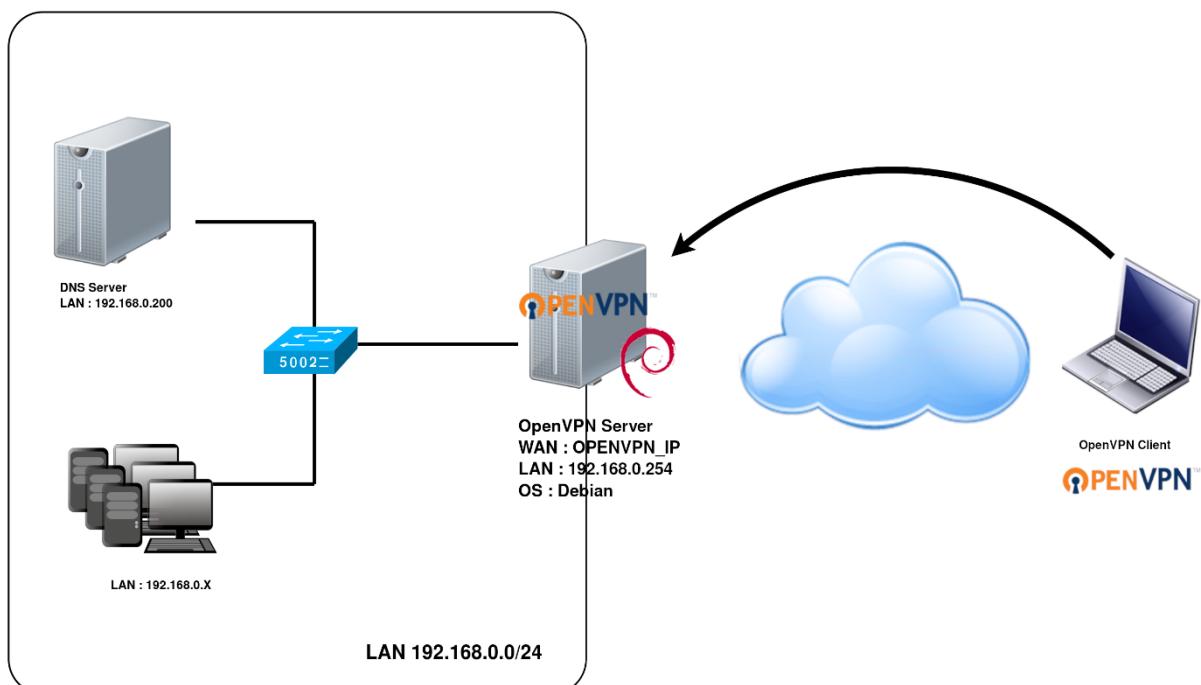
comp-lzo
verb 3

[ Read 16 lines ]
^G Help      ^O Write Out  ^F Where Is  ^K Cut      ^T Execute  ^C Location
^X Exit      ^R Read File  ^\ Replace   ^U Paste    ^J Justify  ^/ Go To Line

```

**Nous avons créé un fichier de configuration client.ovpn qui contient les paramètres nécessaires à la connexion VPN :**

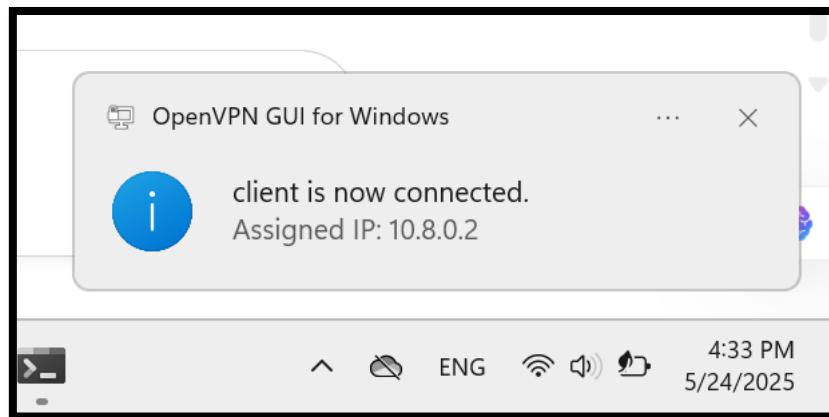
- **Type d'interface (tun)**
- **Protocole (udp)**
- **Adresse IP et port du serveur VPN (exemple : 172.20.10.2 1194)**
- **Chemins relatifs vers les certificats et clés**
- **Options de persistance et de compression**



## Importation dans OpenVPN GUI sous Windows

Le dossier client avec tous les fichiers a été copié dans le répertoire C:\Program Files\OpenVPN\config\ sur Windows.

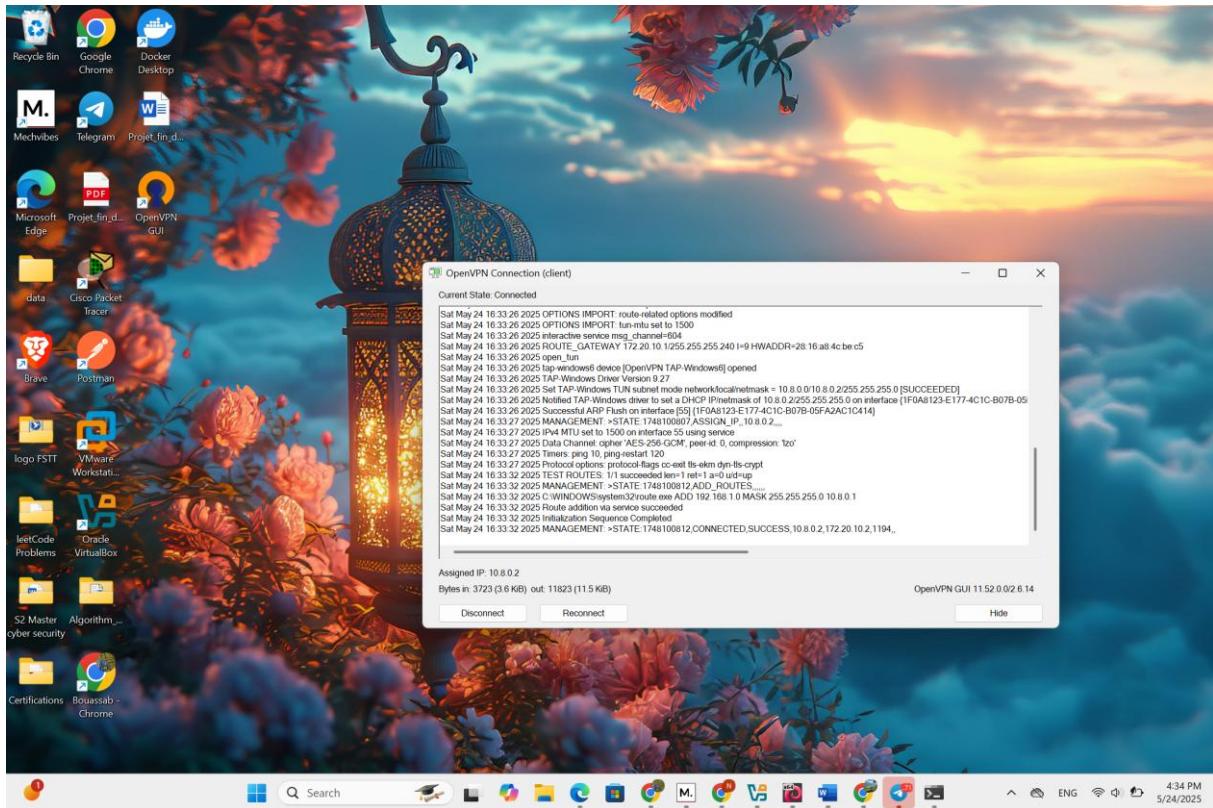
Ensuite, l'application OpenVPN GUI a été lancée en mode administrateur pour permettre la gestion des connexions réseau.



## Connexion et test

### Démarrage de la connexion VPN via OpenVPN GUI

- L'icône OpenVPN est visible dans la barre système (près de l'horloge Windows).
- Un clic droit sur l'icône affiche le profil client.ovpn.
- Nous avons cliqué sur Connect pour démarrer la connexion.



## Analyse des logs et état de la connexion

La fenêtre de log montre l'état détaillé de la connexion. Nous avons vérifié :

- L'authentification des certificats
- L'établissement de la connexion UDP au serveur
- L'attribution d'une adresse IP VPN (exemple : 10.8.0.2)
- L'absence d'erreur critique ou d'avertissement important

```
C:\Users\surface>ping 10.8.0.1

Pinging 10.8.0.1 with 32 bytes of data:
Reply from 10.8.0.1: bytes=32 time=2ms TTL=64
Reply from 10.8.0.1: bytes=32 time=2ms TTL=64
Reply from 10.8.0.1: bytes=32 time=2ms TTL=64
Reply from 10.8.0.1: bytes=32 time=22ms TTL=64

Ping statistics for 10.8.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 22ms, Average = 7ms

C:\Users\surface>
```

Le client VPN (Windows) communique correctement avec le serveur VPN (Kali)

Le tunnel VPN est actif, sécurisé et bidirectionnel

L'adresse 10.8.0.1 est bien joignable à travers le tunnel VPN, donc :

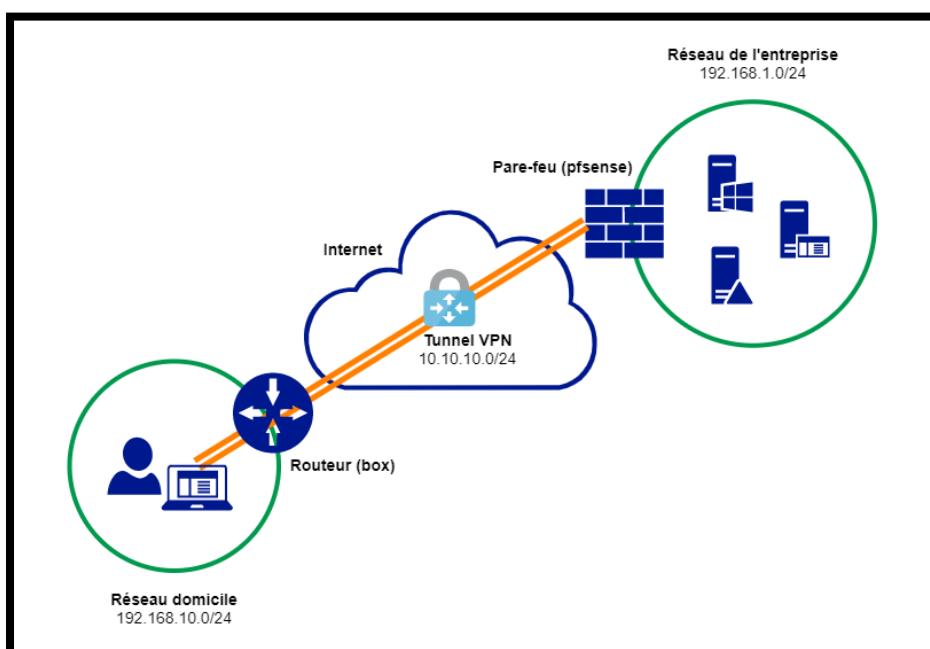
- La connexion est chiffrée
- Les configurations server.conf et client.ovpn sont valides
- La routabilité entre client et serveur est assurée

La configuration d'un tunnel VPN sécurisé a été réalisée avec succès entre le serveur OpenVPN installé sur une machine Kali Linux et un client Windows à l'aide d'OpenVPN GUI.

Après avoir généré et transféré les certificats nécessaires (client.ovpn, ca.crt, client1.crt, client1.key, ta.key), la connexion a été établie correctement.

La capture d'écran ci-dessus confirme que :

- Le client s'est vu attribuer l'adresse VPN 10.8.0.2
- Le serveur est joignable à l'adresse 10.8.0.1
- L'état de connexion est "CONNECTED, SUCCESS"
- Le message "Initialization Sequence Completed" confirme que le tunnel est pleinement opérationnel



## **4. Conclusion générale**

### **Résumé des résultats**

- La mise en place du VPN OpenVPN a été réalisée avec succès. Le serveur est opérationnel, les certificats ont été correctement générés et transférés, et la configuration client a permis d'établir une connexion sécurisée entre le client Windows et le serveur Kali Linux.
- Confirmation de la connexion sécurisée
- Les échanges sont protégés par un chiffrement SSL/TLS, garantissant la confidentialité et l'intégrité des données transitant par le tunnel VPN. Les tests de ping confirment la communication fonctionnelle.

### **Suggestions d'amélioration ou étapes futures possibles**

- Mise en place d'une gestion dynamique des utilisateurs avec authentification par mot de passe en plus des certificats
- Automatisation des scripts de configuration et déploiement
- Ajout d'un serveur de logs centralisé pour la supervision des connexions
- Mise en place de règles de pare-feu renforcées pour sécuriser davantage le serveur VPN
- Tester le VPN sur différents réseaux (WiFi public, 4G) pour valider la robustesse