

PROJECT SILENT BRIDGE

The aim of the project is to explore how a single machine can be transformed into a dual layer Private Network Gateway by deploying for both WireGuard and OpenVPN side by side. The goal is to build a fast, resilient and stealth-capable secure communication tunnel while ensuring we have a grasp of modern VPN architecture. Each step from system preparation to traffic routing will reveal how these technologies behave, interact and complement one another. The project not only encompasses setting up a VPN software but also shaping the machine into a versatile security node that will combine speed, robustness and anonymity while using the internet.

Architecture Overview

Dual-VPN Gateway Concept

The system hosts two independent VPN environments:

WireGuard

Subnet: 10.10.0.0/24

Port: 51820/udp

OpenVPN

Subnet: 10.8.0.0/24

Port: 443/tcp (stealth under HTTPS-like traffic)

Why Two VPNs?

Clean routing separation

Avoid overlapping routes

Easier debugging

Stealth operation in restrictive networks

Faster WireGuard performance + mature OpenVPN PKI stack

Phase one – System Preparations.

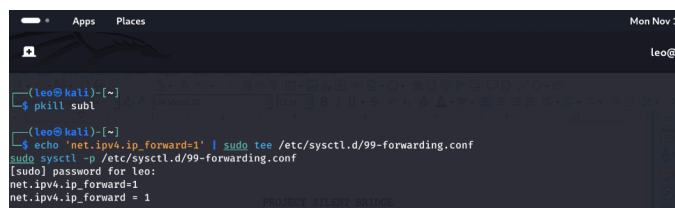
Your machine becomes a router.

To ensure for a stable and consistent environment the first step is always running a system update and upgrade. In Kali distro, the command `sudo apt update && sudo apt full-upgrade -y` should do it. The `-y` ensures that the system automatically answers yes to the prompts of if we wish to update... with that set we are guaranteed that the WireGuard, OpenVPN and the kernel-level networking components operate using the most recent stable versions.

Enabling IPv4 forwarding.

This transforms the Kali machine into a gateway capable of routing the VPN traffic to external network. This step elevates the system from standard host to functional network router. To achieve this the command :

`echo 'net.ipv4.ip_forward=1' | sudo tee /etc/sysctl.d/99-forwarding.conf`
`sudo sysctl -p /etc/sysctl.d/99-forwarding.conf` is run.

A terminal window screenshot from a Kali Linux machine. The prompt is (leo@kali)-[~]. The user enters `$ kill subl`. Then they enter `$ echo 'net.ipv4.ip_forward=1' | sudo tee /etc/sysctl.d/99-forwarding.conf`. The terminal shows the output of the `tee` command as `net.ipv4.ip_forward=1`. Next, they enter `$ sudo sysctl -p /etc/sysctl.d/99-forwarding.conf`. The terminal shows the output of the `sysctl` command as `net.ipv4.ip_forward = 1`. The window title bar shows 'Apps', 'Places', and 'Mon Nov 1'. The user's name 'leo@' is visible in the top right corner of the terminal window.

```
(leo@kali)-[~]
$ kill subl
$ echo 'net.ipv4.ip_forward=1' | sudo tee /etc/sysctl.d/99-forwarding.conf
net.ipv4.ip_forward=1
$ sudo sysctl -p /etc/sysctl.d/99-forwarding.conf
net.ipv4.ip_forward = 1
```

A successful output should be as **`net.ipv4.ip_forward = 1`**.

install and prepare the firewall:- Uncomplicated Firewall [UFW].

This step keeps the environment controlled by ensuring that OpenVPN and WireGuard get their own clean doorway/port.

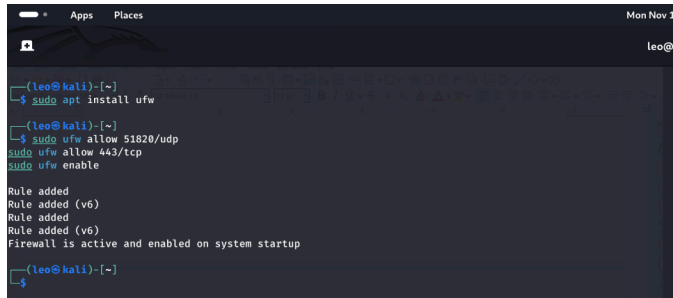
The command `sudo apt install ufw -y`.

our preferred port of choice are port 51820/udp and port 443/tcp. To enable the ports and the firewall run the commands.

```
Sudo apt allow 51820/udp
```

```
sudo ufw allow 443/tcp
```

```
sudo ufw enable
```

A terminal window screenshot showing the installation and configuration of UFW. The commands entered are: `sudo apt install ufw`, `sudo ufw allow 51820/udp`, `sudo ufw allow 443/tcp`, and `sudo ufw enable`. The output shows three rules being added and the firewall being enabled on system startup.

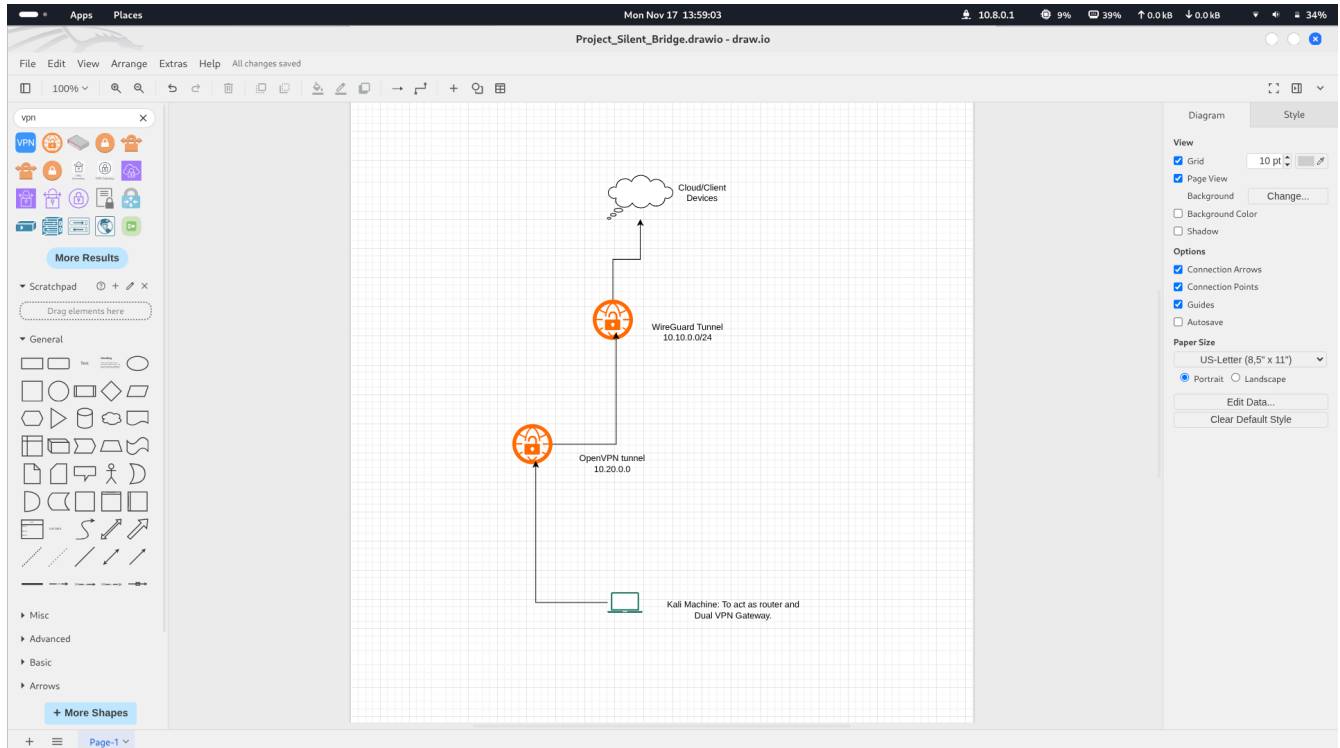
```
(leo@kali)~$ sudo apt install ufw
--(leo@kali)~$ sudo ufw allow 51820/udp
--(leo@kali)~$ sudo ufw allow 443/tcp
--(leo@kali)~$ sudo ufw enable
Rule added
Rule added (v6)
Rule added
Rule added (v6)
Firewall is active and enabled on system startup
--(leo@kali)~$
```

if a port is already running or allowed use the command `sudo ufw delete allow <port/service>`: assuming port 51820/udp is running the command `sudo ufw delete allow 51820/udp`. Then use the command `sudo ufw allow <port/service>` ie `sudo ufw allow 51820/udp`.

The above is set to allow/support clean traffic management with the use of the UFW [Uncomplicated Firewall] was configured to allow inbound VPN connections. Port 51820/udp was reserved for WireGuard while 443/tcp was allocated for OpenVPN for stealth operation over HTTP/S like traffic. The dual port setup ensures that both services can coexist without conflict.

The Network Blueprint.

The project is to use two independent virtual subnets to ensure clean routing separation. WireGuard will operate over the 10.10.0.0/24 network while OpenVPN is assigned the 10.20.1.0/24. this is to avoid overlapping routes as well as simplify debugging and traffic tracing. WireGuard is to listen on port 51820/udp, and OpenVPN is to run over port 443/TCP for improved stealth in restrictive environment.



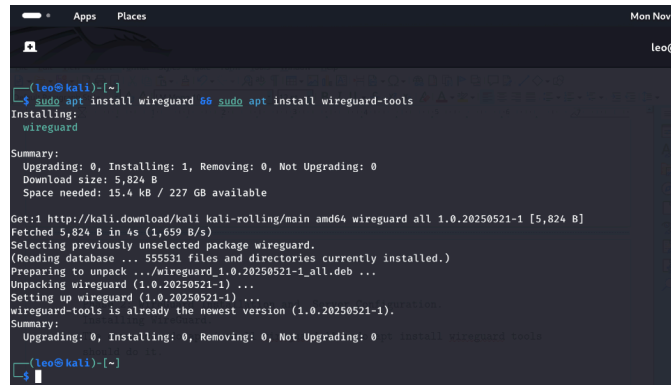
the diagram illustrates a visual map for our expected outcome.

WIREGUARD INSTALLATION AND SETUP

Phase 2: WireGuard Installation and Server Configuration.

Installing WireGuard.

The command `sudo apt install WireGuard` & `sudo apt install WireGuard tools` should do it.



```
(leo@kali)-[~]
└─$ sudo apt install wireguard && sudo apt install wireguard-tools
Installing:
wireguard

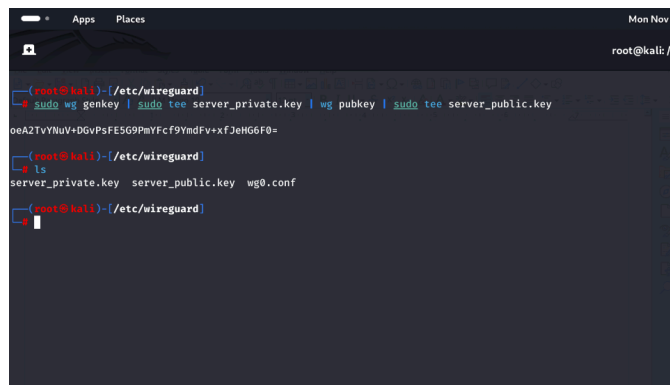
Summary:
  Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 0
  Download size: 5,824 B
  Space needed: 15.4 kB / 227 GB available

Get:1 http://kali.download/kali kali-rolling/main amd64 wireguard all 1.0.20250521-1 [5,824 B]
Fetched 5,824 B in 4s (1,659 B/s)
Selecting previously unselected package wireguard.
(Reading database ... 555531 files and directories currently installed.)
Preparing to unpack .../wireguard_1.0.20250521-1_all.deb ...
Unpacking wireguard (1.0.20250521-1) ...
Setting up wireguard (1.0.20250521-1) ...
wireguard-tools is already the newest version (1.0.20250521-1).
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
└─$ sudo apt install wireguard tools
```

Generating Server Keys.

Navigate to the /etc/wireguard and generate the keys from the directory.

The command **sudo wg genkey | sudo tee server_private.key | wg pubkey | sudo tee server_public.key**



```
root@kali: /etc/wireguard
$ sudo wg genkey | sudo tee server_private.key | wg pubkey | sudo tee server_public.key
oeA2TvYnuV+DgvPsFE5G9PmYFcF9YmdFv+xfJehG6F8=
root@kali: /etc/wireguard
$ ls
server_private.key  server_public.key  wg0.conf
root@kali: /etc/wireguard
```

“WireGuard uses a simple public–private key model. The server’s key pair was generated in /etc/wireguard/, with strict permissions applied to protect the private key. These keys authenticate and encrypt all VPN communication.”

ensure to make the key readable to the root only with the command :

sudo chmod 600 server_private.key

this is done to ensure restrictive permissions as it contains our private key

creating a WireGuard Interface [wg0]

this step is where we build the server configurations.

We start by creating a file named wg0.conf in /etc/wireguard with the command
`sudo nano /etc/wireguard/wg0.conf`

```
root@kali: /etc/wireguard
# sudo wg genkey | sudo tee server_private.key | wg pubkey | sudo tee server_public.key
oeA2TtYNUv+DgVp5FE5G9PmYfc9YmdFv+xfJehG6F0=
root@kali: /etc/wireguard
# ls
server_private.key  server_public.key  wg0.conf
root@kali: /etc/wireguard
# sudo chmod 600 server_private.key

root@kali: /etc/wireguard
# ls -la
total 28
drwxr-xr-x  2 root root 4096 Nov 17 14:38 .
drwxr-xr-x 288 root root 12288 Nov 17 13:26 ..
-rw-r--r--  1 root root  45 Nov 17 14:38 server_private.key
-rw-r--r--  1 root root  45 Nov 17 14:38 server_public.key
-rw-r--r--  1 root root 246 Nov 10 13:27 wg0.conf
root@kali: /etc/wireguard
# sudo subl wg0.conf

root@kali: /etc/wireguard
# cat server_private.key
qPZTbdJvfJ14MaoHEH//3TxkKtXxKsckdg69jBmH9H8=
root@kali: /etc/wireguard
# cat server_private.key
qPZTbdJvfJ14MaoHEH//3TxkKtXxKsckdg69jBmH9H8=

root@kali: /etc/wireguard
# cat wg0.conf
[Interface]
Address = 10.10.0.1/24
ListenPort = 51820
PrivateKey = qPZTbdJvfJ14MaoHEH//3TxkKtXxKsckdg69jBmH9H8=

# NAT rules for outbound traffic
PostUp = iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
PostDown = iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE
root@kali: /etc/wireguard
```

in the wg0.conf insert

[Interface]

Address = 10.10.0.1/24

ListenPort = 51820

PrivateKey = <SERVER_PRIVATE_KEY>

NAT rules for outbound traffic

PostUp = iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

PostDown = iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE

“The primary WireGuard interface configuration (wg0) defines the server’s VPN IP, port, and routing rules. NAT masquerading is enabled through PostUp and PostDown directives, allowing VPN clients to reach external networks through the server’s primary interface.”

Start and Enable WireGuard interface.

The command

```
sudo wg-quick up wg0
```

```
leo@kali:~$ sudo wg-quick up wg0
[sudo] password for leo:
[#] ip link add dev wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 10.10.0.1/24 dev wg0
[#] ip link set mtu 1420 up dev wg0
[#] iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

which starts WireGuard.

To enable on startup

```
sudo systemctl enable wg-quick@wg0
```

```
leo@kali:~$ sudo systemctl enable wg-quick@wg0
Created symlink '/etc/systemd/system/multi-user.target.wants/wg-quick@wg0.service' → '/usr/lib/systemd/system/wg-quick@.service'.
```

to view status

```
sudo wg
```

where you should see something like.

```
interface: wg0
public key: ABC123...
private key: (hidden)
listening port: 51820
```

```
leo@kali:~$ sudo wg
interface: wg0
public key: oA2TYVYNgV+DGvPsFE5G9PmYFcF9YndFv+xfJeHG6F8=
private key: (hidden)
listening port: 51820
```

the WireGuard interface is activated in the command `sudo wg-quick up wg0` and hence turning the configuration into a live VPN endpoint. And the command `sudo systemctl enable wg-quick@wg0` ensures that the WireGuard starts on system startup/boot.

Phase 3: Generate Client keys

this phase ensures for
a working client keypair

- a client IP on the VPN subnet

- a full client.conf you can use on another device

- server ↔ client handshake verification

Generating client keys.

Generating for client private and public keys with the command

wg genkey | tee client1_privatekey | wg pubkey > client1_publickey

```
(root@kali)~# /etc/wireguard
# wg genkey | tee client1_privatekey | wg pubkey > client1_publickey
(root@kali)~# /etc/wireguard
# ls
client1_privatekey  server_private.key  wg0.conf
client1_publickey  server_public.key
```

a dedicate key pair is generated for the WireGuard client. This is because WireGuard assigns identities based on key pairs thus enabling lightweight built and secure authentication.

Add the client as the peer to the server

in the

```
(root@kali)~# /etc/wireguard
# cat wg0.conf
[Interface]
Address = 10.10.0.1/24
ListenPort = 51820
PrivateKey = mNu/xBL6ThMKKn5VyKEnvdW6Rz8hy0mOkcQSPgSnNGA=

PostUp = iptables -t nat -A POSTROUTING -s 10.10.0.0/24 -o wlan0 -j MASQUERADE
PostDown = iptables -t nat -D POSTROUTING -s 10.10.0.0/24 -o wlan0 -j MASQUERADE

[Peer]
PublicKey = 31GR/CPqs/sOYQQZEJqoVH1msWN336o+fn6xufMsKBU=
AllowedIPs = 10.10.0.2/32
```

/etc/wireguard/wg0.conf add the lines below the PostDown Section

```
[Peer]
PublicKey = <client1_publickey>
AllowedIPs = 10.10.0.2/32
```

ensure to replace your client key with the actual contents of your client key you generated.
The peer section prepares the server to accept our client.

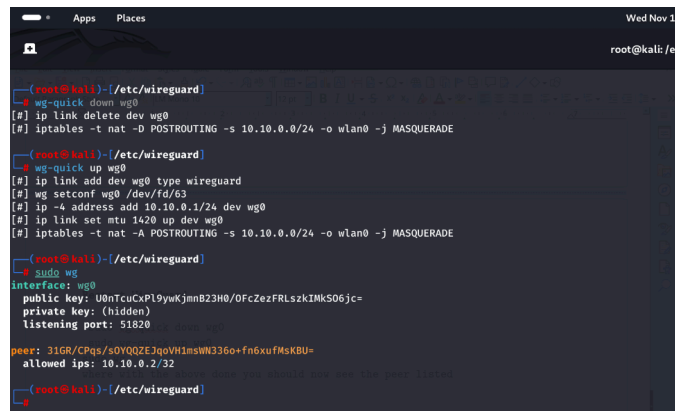
Restart WireGuard

sudo wg-quick down wg0

sudo wg-quick up wg0

sudo wg

where with the above done you should now see the peer listed

A terminal window screenshot showing the process of restarting WireGuard. The user is in the root directory of a Kali Linux machine. They first run 'wg-quick down wg0' and then 'wg-quick up wg0'. The terminal shows the underlying commands being executed, including 'ip link delete dev wg0', 'wg setconf /dev/fd/63', 'ip -4 address add 10.10.0.1/24 dev wg0', 'ip link set mtu 1420 up dev wg0', and 'iptables -t nat -A POSTROUTING -s 10.10.0.0/24 -o wlan0 -j MASQUERADE'. After running 'sudo wg', the terminal displays the configuration for the 'wg0' interface, including the public key 'U0nTcuCxPl9yWkjmB23H0/0FcZezFRLszkIMkS06jc=' and the listening port '51820'. It also shows a peer with the public key '31GR/CPqs/s0YQQZEJqoVH1msWN336o+fn6xufMsKBU=' and allowed IPs '10.10.0.2/32'.

```
(root@kali): /etc/wireguard
# wg-quick down wg0
[#] ip link delete dev wg0
[#] iptables -t nat -D POSTROUTING -s 10.10.0.0/24 -o wlan0 -j MASQUERADE

(root@kali): /etc/wireguard
# wg-quick up wg0
[#] ip link add dev wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 10.10.0.1/24 dev wg0
[#] ip link set mtu 1420 up dev wg0
[#] iptables -t nat -A POSTROUTING -s 10.10.0.0/24 -o wlan0 -j MASQUERADE

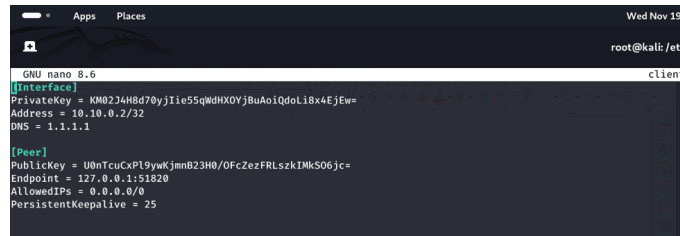
(root@kali): /etc/wireguard
# sudo wg
interface: wg0
  public key: U0nTcuCxPl9yWkjmB23H0/0FcZezFRLszkIMkS06jc=
  private key: (hidden)
  listening port: 51820

peer: 31GR/CPqs/s0YQQZEJqoVH1msWN336o+fn6xufMsKBU=
  allowed ips: 10.10.0.2/32
  (Note you should now see the peer listed)

(root@kali): /etc/wireguard
```

create a client configuration file

with the command **sudo nano client1.conf**



```
GNU nano 8.6 client
[Interface]
PrivateKey = KM02J4H8d70yjiE55qWdHXOYjBuAoiQd0Li8x4EjEw=
Address = 10.10.0.2/32
DNS = 1.1.1.1

[Peer]
PublicKey = U0nTcuCpL9ywKjmmB23H0/OfcZezFRLszkIMkS06jc=
Endpoint = 127.0.0.1:51820
AllowedIPs = 0.0.0.0/0
PersistentKeepalive = 25
```

in the .conf paste for

[Interface]

PrivateKey = <client1_privatekey>

Address = 10.10.0.2/32

DNS = 1.1.1.1

[Peer]

PublicKey = <server_publickey>

Endpoint = <SERVER_PUBLIC_IP>:51820

AllowedIPs = 0.0.0.0/0

PersistentKeepalive = 25

and replace where relevant ie with <...>. Since the server will be running on your local kali machine fill the endpoint IP address as 127.0.0.1 which is you loopback IP address.

=> Ensure to run the command `sudo systemctl enable wg-quick@client1` to start it on bootup

Starting the client

the command `sudo wg-quick up client1` does it

```
➤ chmod 600 /etc/wireguard/client1.conf

(root@kali) ~ - [etc/wireguard]
# wg-quick up client1
[#] ip link add dev client1 type wireguard
[#] wg setconf client1 /dev/fd/63
[#] ip -4 address add 10.10.0.2/32 dev client1
[#] ip link set mtu 65456 up dev client1
[#] resolvconf -a tun.client1 -m 0 -x
[#] wg set client1 fwmark 51820
[#] ip -4 rule add not fwmark 51820 table 51820
[#] ip -4 rule add table main suppress_prefixlength 0
[#] ip -4 route add 0.0.0.0/0 dev client1 table 51820
[#] sysctl -q net.ipv4.conf.all.src_valid_mark=1
[#] nft -f /dev/fd/63

(root@kali) ~ - [etc/wireguard]
```

if faced by `resolveconf`

`resoleconf`: command not found

1. install resolve with
`sudo apt install resolveconf -y`
2. then try `sudo wg-quick up client1`

INSTALLING AND SETTING UP OPENVPN.

Update the system.

Good kali practice as mentioned is always updating and upgrading your system before any installation. As mentioned the command: **sudo apt update** updates the system while the command: **sudo apt upgrade** upgrades the system. Or combine the commands to: **sudo apt update && sudo apt full-upgrade -y**

this practice ensures that all packages which in our case are [OpenVPN, Easy-SRA, Kernel, Networking tools] are in the latest stable versions.

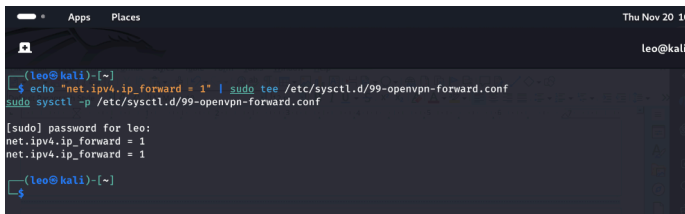
Enabling IP forwarding.

To enable IP forwarding the command:

```
echo "net.ipv4.ip_forward = 1" | sudo tee /etc/sysctl.d/99-openvpn-forward.conf
```

```
sudo sysctl -p /etc/sysctl.d/99-openvpn-forward.conf
```

this allows our server to route IP traffic between our client and the internet.

A screenshot of a Kali Linux terminal window. The window title bar shows 'Apps', 'Places', and the date 'Thu Nov 20 14:00'. The terminal shows the user 'leo@kali:~' at the prompt. The user enters the command `echo "net.ipv4.ip_forward = 1" | sudo tee /etc/sysctl.d/99-openvpn-forward.conf`. The prompt changes to `[sudo]` and the user enters their password. The terminal then displays the output of the command: `net.ipv4.ip_forward = 1` and `net.ipv4.ip_forward = 1`. The prompt returns to `leo@kali:~`.

the expected output on running the command: **sysctl -p ...** should be `net.ipv4.ip_forward = 1`.

Set up Network Address Translator Table/Iptables

the command:

```
sudo iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o wlan0 -j MASQUERADE
```

should do it. "A NAT masquerading rule was added so that OpenVPN client traffic (from the 10.8.0.0/24 subnet) exits via the server's wlan0 interface, hiding client IPs behind the server."

to save the rules to the
them persistent use

```
sudo apt install  
sudo  
save
```

this ensures that the
across reboot.

```
(leo@kali)-[~]  
└─$ sudo iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o wlan0 -j MASQUERADE  
  
(leo@kali)-[~]  
└─$ sudo iptables -t nat -L -n -V  
iptables v1.8.11 (nf_tables)  
  
(leo@kali)-[~]  
└─$ sudo iptables -t nat -L -n -V  
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)  
pkts bytes target prot opt in out source destination  
  
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)  
pkts bytes target prot opt in out source destination  
  
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)  
pkts bytes target prot opt in out source destination  
  
Chain POSTROUTING (policy ACCEPT 9021 packets, 1953K bytes)  
pkts bytes target prot opt in out source destination  
9017 1953K amvnpn.anchors all -- * * 0.0.0.0/0 0.0.0.0/0  
0 0 MASQUERADE all -- * wlan0 10.0.0.0/8 0.0.0.0/0  
0 0 MASQUERADE all -- * wlan0 10.8.0.0/24 0.0.0.0/0  
0 0 MASQUERADE all -- * wlan0 10.8.0.0/24 0.0.0.0/0  
  
Chain amvnpn.100.transIp (0 references)  
pkts bytes target prot opt in out source destination  
0 0 MASQUERADE all -- * * 0.0.0.0/0 0.0.0.0/0  
  
Chain amvnpn.a.100.transIp (1 references)  
pkts bytes target prot opt in out source destination  
  
Chain amvnpn.anchors (1 references)  
pkts bytes target prot opt in out source destination  
9017 1953K amvnpn.a.100.transIp all -- * * 0.0.0.0/0 0.0.0.0/0  
  
(leo@kali)-[~]  
└─$
```

IP tables and make
the command.

iptables-persistent
netfilter-persistent

NAT rule persistent

Install OpenVPN and Easy-RSA

to install OpenVpn run the command

```
sudo apt install openvpn easy-rsa -y
```

```

Sun Nov 23 19:29:05
leo@kali: ~

leo@kali:~$ sudo apt install openvpn easy-rsa -y
[sudo] password for leo:
easy-rsa is already the newest version (3.2.4-1).
The following package was automatically installed and is no longer required:
  libportugol0
Use 'sudo apt autoremove' to remove it.

Installing:
  openvpn

Installing dependencies:
  libpks11-helper1t64

Suggested packages:
  openvpn-dco-dkms  openvpn-systemd-resolved

Summary:
  Upgrading: 0, Installing: 2, Removing: 0, Not Upgrading: 0
  Download size: 719 kB
  Space needed: 1,997 kB / 225 GB available

Get:1 http://http.kali.org/kali kali-rolling/main amd64 libpks11-helper1t64 amd64 1.30.0-1+b1 [51.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 openvpn amd64 2.6.15-1 [667 kB]
Fetched 719 kB in 8s (93.1 kB/s)
Preconfiguring packages ...
Selecting previously unselected package libpks11-helper1t64:amd64.
(Reading database ... 555446 files and directories currently installed.)
Preparing to unpack .../libpks11-helper1t64_1.30.0-1+b1_amd64.deb ...
Unpacking libpks11-helper1t64:amd64 (1.30.0-1+b1) ...
Selecting previously unselected package openvpn.
Preparing to unpack .../openvpn_2.6.15-1_amd64.deb ...
Unpacking openvpn (2.6.15-1) ...
Setting up libpks11-helper1t64:amd64 (1.30.0-1+b1) ...
Setting up openvpn (2.6.15-1) ...
update-rc.d: We have no instructions for the openvpn init script.
update-rc.d: It looks like a network service, we disable it.
openvpn.service is a disabled or a static unit, not starting it.
Processing triggers for man-db (2.13.1-1) ...
Processing triggers for kali-menu (2025.4.2) ...
Processing triggers for libc-bin (2.41-12) ...

leo@kali:~$
```

Step 2: Set Up the Certificate Authority (CA)

1. Create a directory for Easy-RSA:

```
(leo@kali)-[~]
$ make-cadir ~/openvpn-ca
cd ~/openvpn-ca

(leo@kali)-[/openvpn-ca]
$ ls
easyrsa openssl-easyrsa.cnf vars x509-types
$
```

the commands above creates the directory and navigates to it.

- ## 2. Initialize the PKI

```

leo@kali: ~/openvpn-ca
└─$ ./easyrsa init-pki
Using Easy-RSA 'vars' configuration:
* /home/leo/openvpn-ca/vars

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

Your newly created PKI dir is:
* /home/leo/openvpn-ca/pki

Using Easy-RSA configuration:
* /home/leo/openvpn-ca/vars

leo@kali: ~/openvpn-ca
└─$

```

- ### 3. Build a Certificate Authority

[illegible]

Easy-RSA's PKI (Public Key Infrastructure) was initialized. A root CA certificate was generated ('MyVPN-CA') without a password to simplify automation, but private keys are stored securely. The nopass means no password hence making automation simpler but should be secure in an ideal setup.

Generate the server Certificate.

[illegible]

server is the name of the certificate.

nopass makes it unencrypted for startup ease.

A server certificate and private key were generated using Easy-RSA. These form the identity of our OpenVPN server.

Generating the Diffie-Hellman Parameter

```
[leo@kali]~$ cd /etc/openvpn-ca
[leo@kali]~/etc/openvpn-ca$ ./easyrsa gen-dh
Using Easy-RSA 'vars' configuration:
# /home/leo/openvpn-ca/vars
Generating DH parameters, 2048 bit long safe prime
.....
```

the Diffie-Hellman DH is generated to ensure for a secure key exchange between the clients and the server.

Copy the materials to the OpenVPN directory in the /etc directory with the commands

```
sudo cp ~/openvpn-ca/pki/ca.crt /etc/openvpn/
```

```
sudo cp ~/openvpn-ca/pki/issued/server.crt /etc/openvpn/
```

```
sudo cp ~/openvpn-ca/pki/private/server.key /etc/openvpn/
```

```
sudo cp ~/openvpn-ca/pki/dh.pem /etc/openvpn/
```

All

```
(leo@kali)~[/openvpn-ca]
$ ls
easyrsa  openssl-easyrsa.cnf  pki  vars  x509-types

(leo@kali)~[/openvpn-ca]
$ sudo cp ~/openvpn-ca/pki/ca.crt /etc/openvpn/
$ sudo cp ~/openvpn-ca/pki/issued/server.crt /etc/openvpn/
$ sudo cp ~/openvpn-ca/pki/private/server.key /etc/openvpn/
$ sudo cp ~/openvpn-ca/pki/dh.pem /etc/openvpn/

[sudo] password for leo:
(leo@kali)~[/openvpn-ca]
$ sudo ls /etc/openvpn
ca.crt  client  dh.pem  server  server.crt  server.key  update-resolv-conf

(leo@kali)~[/openvpn-ca]
$
```

required cryptographic files (CA certificate, server certificate, private key, DH parameters) were copied to the OpenVPN directory for use by the service.

Generate the OpenVPN server configurations.

In the `openvpn` directory create a directory named `server` and a file named `server.conf` as per the image below.

```
(leo@kali) ~/openvpn-ca
└─$ sudo mkdir -p /etc/openvpn/server
sudo nano /etc/openvpn/server/server.conf

┌─(leo@kali)~/openvpn-ca
└─$ sudo cat /etc/openvpn/server/server.conf
port 1194
proto udp
dev tun

ca /etc/openvpn/ca.crt
cert /etc/openvpn/server.crt
key /etc/openvpn/server.key
dh /etc/openvpn/dh.pem

topology subnet
server 10.8.0.0 255.255.255.0

push "redirect-gateway def1"
push "dhcp-option DNS 1.1.1.1"

keepalive 10 120
persist-key
persist-tun

user nobody
group nogroup

verb 3

┌─(leo@kali)~/openvpn-ca
└─$
```

In the server.conf paste the the code

```
port 1194
```

```
proto udp
```

```
dev tun
```

```
ca /etc/openvpn/ca.crt
```

```
cert /etc/openvpn/server.crt
```

```
key /etc/openvpn/server.key
```

```
dh /etc/openvpn/dh.pem
```

```
topology subnet
```

```
server 10.8.0.0 255.255.255.0
```

```
push "redirect-gateway def1"
```

```
push "dhcp-option DNS 1.1.1.1"
```

```
keepalive 10 120
```

```
persist-key
```

```
persist-tun
```

```
user nobody
```

```
group nogroup
```

```
verb 3
```

start and enable the openvpn

to

```
(leo@kali) [/openvpn-ca]
$ sudo systemctl start openvpn-server@server
$ sudo systemctl enable openvpn-server@server

(too) (leo@kali) [/openvpn-ca]
$ sudo systemctl start openvpn-server@server
$ sudo systemctl status openvpn-server@server

● openvpn-server@server.service - OpenVPN service for server
   Loaded: loaded (/usr/lib/systemd/system/openvpn-server@server.service; enabled; preset: disabled)
   Active: active (running) since Sun 2025-11-23 19:49:59 EAT; 18s ago
   Invocation: 1b1f423bdf5d47c68b0a6213b9ebcb6a
   Docs: man:openvpn(8)
         https://openvpn.net/community-resources/reference-manual-for-openvpn-2-6/
         https://community.openvpn.net/openvpn/wiki/HOWTO
   Main PID: 54409 (openvpn)
   Status: "Initialization Sequence Completed"
   Tasks: 1 (limit: 10401)
   Memory: 1.8M (peak: 2.2M)
   CPU: 56ms
   CGroup: /system.slice/system-openvpn\x2dserverslice/openvpn-server@server.service
           └─54409 /usr/sbin/openvpn --status /run/openvpn-server/status-server.log --status-version 2 --suppress-timestamps --config server.conf

Nov 23 19:49:59 kali openvpn[54409]: Could not determine IPv4/IPv6 protocol. Using AF_INET
Nov 23 19:49:59 kali openvpn[54409]: Socket Buffers: R=[212992->212992] S=[212992->212992]
Nov 23 19:49:59 kali openvpn[54409]: UDPv4 link local (bound): [AF_INET][undef]:1194
Nov 23 19:49:59 kali openvpn[54409]: UDPv4 link remote: [AF_UNSPEC]
Nov 23 19:49:59 kali openvpn[54409]: UID set to nobody
Nov 23 19:49:59 kali openvpn[54409]: GID set to nogroup
Nov 23 19:49:59 kali openvpn[54409]: Capabilities retained: CAP_NET_ADMIN
Nov 23 19:49:59 kali openvpn[54409]: MULTI: multi init called, r=256 v=256
Nov 23 19:49:59 kali openvpn[54409]: IFCONFIG POOL: IPv4: base=10.8.0.2 size=253
Nov 23 19:49:59 kali openvpn[54409]: Initialization Sequence Completed

(lleo@kali) [/openvpn-ca]
$
```

start and enable run the commands

`sudo systemctl start openvpn-server@server`

`sudo systemctl enable openvpn-server@server`

and the command : `sudo systemctl status openvpn-server@server` to confirm

conformation.

If done correctly:

or running **ip a** command the results should be as

a

```
(root@kali)~# 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
    inet ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether fc:77:74:de:2d:03 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.107/24 brd 192.168.1.255 scope global dynamic noprefixroute wlan0
        valid_lft 6556sec preferred_lft 6556sec
    inet6 fe80::fe77:74ff:fe0e:2d03/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
6: 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 brd 10.8.0.255 scope global tun0
        valid_lft forever preferred_lft forever
    inet6 fe80::f694:5a21:5978:fd55/64 scope link stable-privacy proto kernel_ll
        valid_lft forever preferred_lft forever
7: client1: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1380 qdisc noqueue state UNKNOWN group default qlen 1000
    link/none
    inet 10.10.0.2/32 scope global client1
        valid_lft forever preferred_lft forever
8: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1420 qdisc noqueue state UNKNOWN group default qlen 1000
    link/none
    inet 10.10.0.1/24 scope global wg0
        valid_lft forever preferred_lft forever
(root@kali)~#
```

tun0 for our OpenVPN, a wgo and client for our wireguard

and from the notification bar



Final System Summary

Once all components are active:

The system acts as a **dual-stack VPN router**

WireGuard provides fast encrypted tunneling

OpenVPN provides certificate-based PKI security

UFW enforces clean separation of services

Both VPNs coexist without port or route conflict

IPv4 forwarding allows full gateway functionality