PROJECT SILENT BRIDGE

The aim of the project is to explore how a 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 built 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 this technologies behave, interact and compliment one another. The project not only encompass 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.

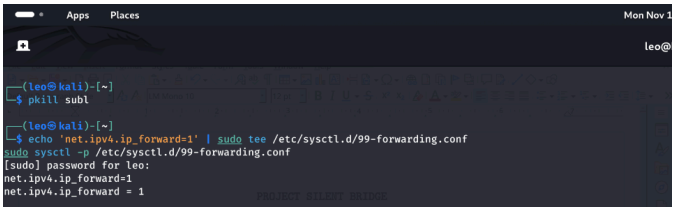
Your machine becomes a router.

To ensure for a stable and consistent environment the firsts tep is always running a system update and upgrade. in kali distro, the command sudo apt update && sudo apt full-upgrade -y should doit. 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 successful output should be as **net.ipv4.ip_forward = 1**.

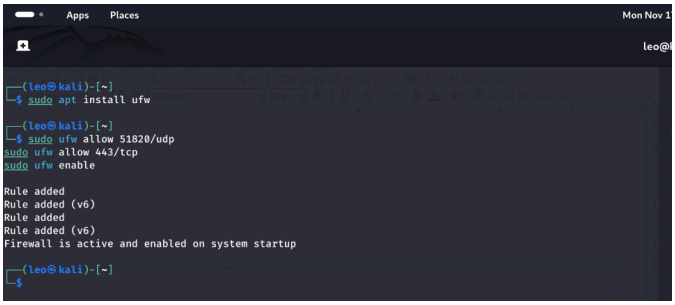
install and prepare the firewall:- Uncomplicated Firewall [UFW].
This step keeps the environment controlled by ensuring that OpenVPN and WireGuard get their own clean doorway/port.

The command sudo apt install ufw -y.
our preferred port of choice are port 51820/udp and port 444/tcp. To enable the ports and the firewall run the commands.

Sudo apt allow 51828/udp

sudo ufw allow 443/tcp

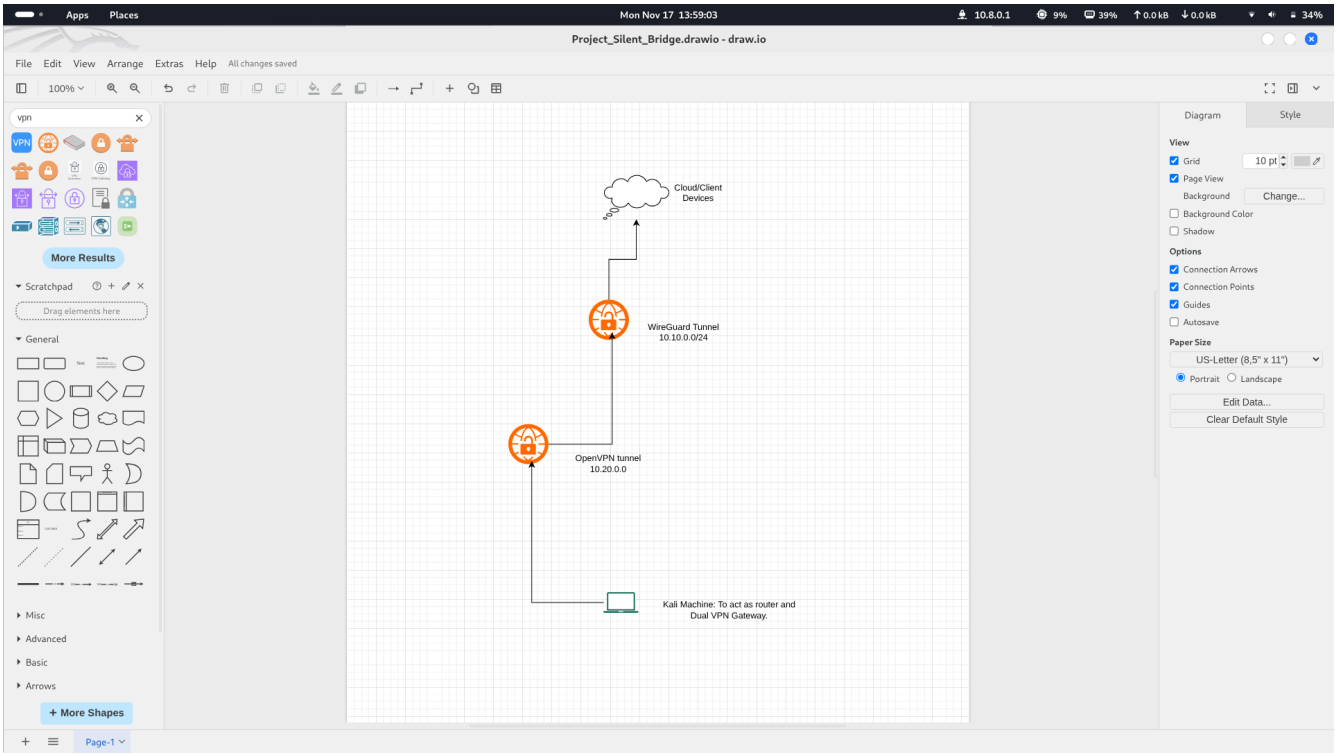
sudo ufw enable



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.



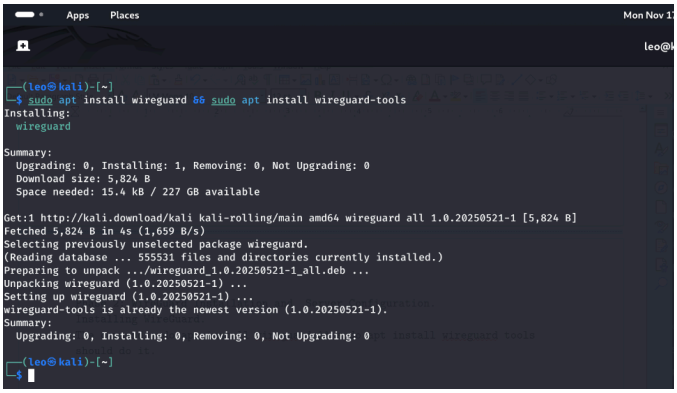
the diagram illustrates a visual map for our expected outcome.

WIREGUARD INSTALLATION AND SETUP

Phase 2: WireGuard Installation and Server Configuration.

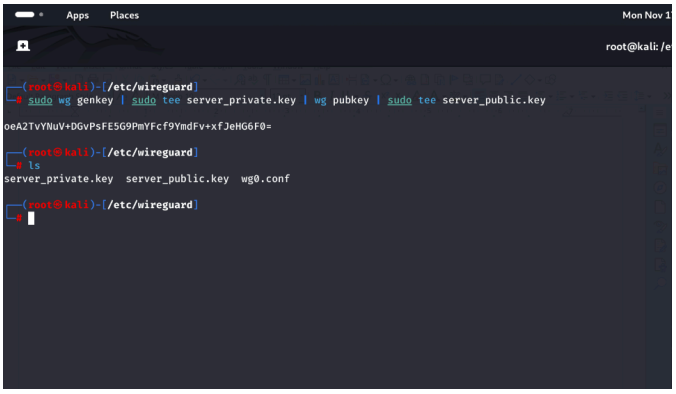
Installing WireGuard.

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



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**



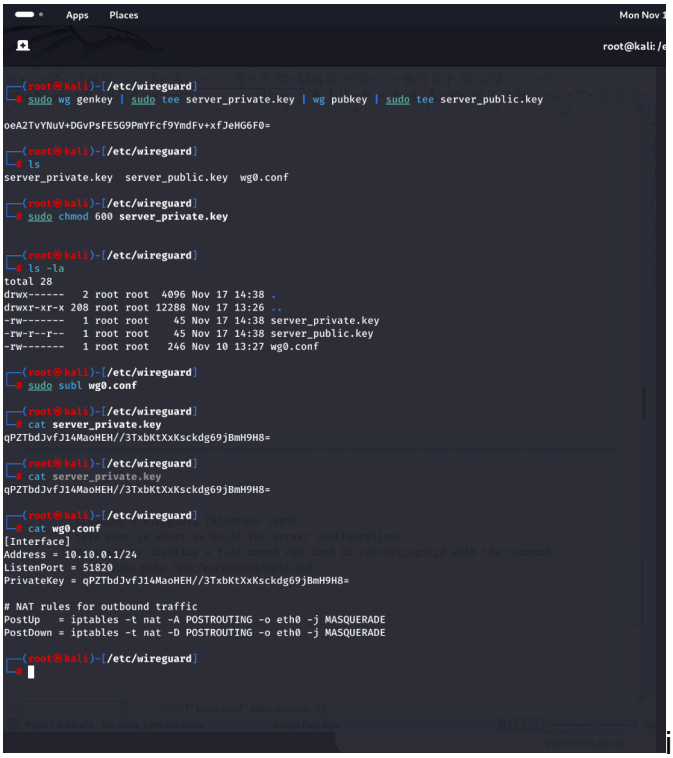
“WireGuard uses a simple public–private key model. The server’s key pair was generated in /etc/wireguard/, with strict permissions applied to protect the private key. These keys authenticate and encrypt all VPN communication.”

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



in the wg0.conf inert

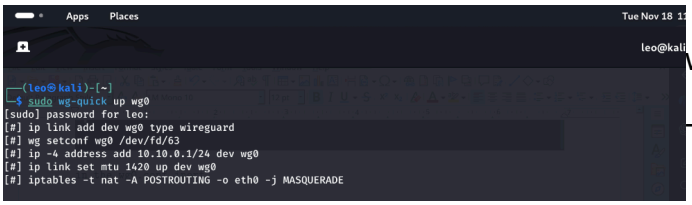
[Interface]
Address = 10.10.0.1/24
ListenPort = 51820
PrivateKey = <SERVER_PRIVATE_KEY>

# 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



which starts WireGuard.

To enable on bootup

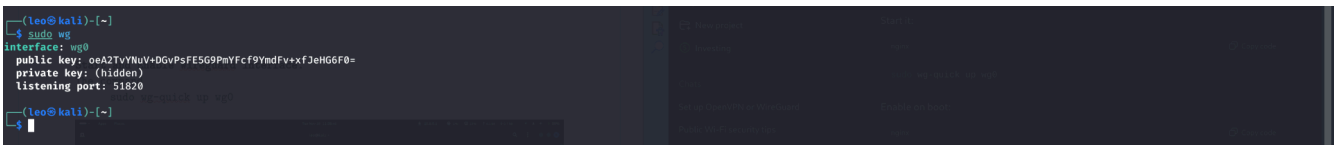
sudo systemctl enable wg-quick@wg0

to view status

sudo wg

where you should see something like.

interface: wg0
public key: ABC123...
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

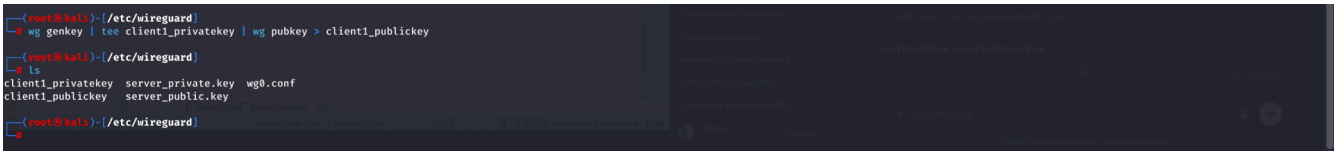
a client IP on the VPN subnet

a full client.conf you can use on another device

server ↔ client handshake verification

Generating client keys.

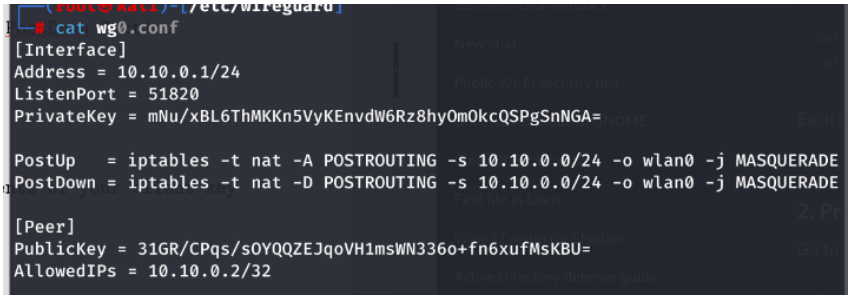
Generating for client privaye and public keys with the command
wg genkey | tee client1_privatekey | wg pubkey > client1_publickey



a dedicate key pair is generated for the WireGuard client. This is becauce 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

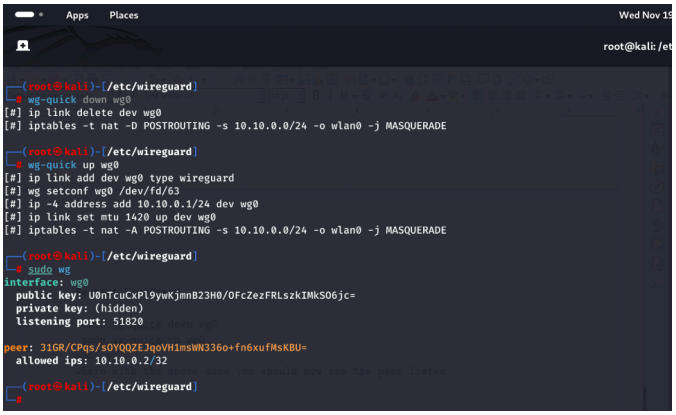


/etc/wireguard/wg0.conf add the lines below the PostDown Section
[Peer]
PublicKey = <client1_publickey>
AllowedIPs = 10.10.0.2/32

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.

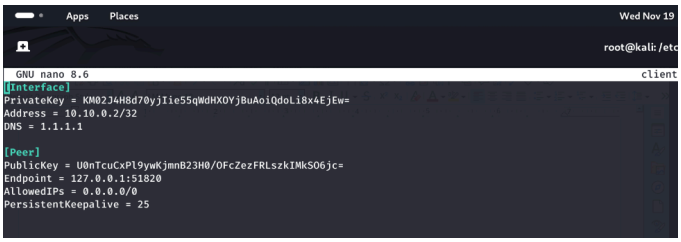
Restart WireGuard

sudo wg-quick down wg0
sudo wg-quick up wg0
sudo wg
where with the above done you should now see the peer listed



create a client configuration file

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



in the .conf paste for

[Interface]
PrivateKey = <client1_privatekey>
Address = 10.10.0.2/32
DNS = 1.1.1.1

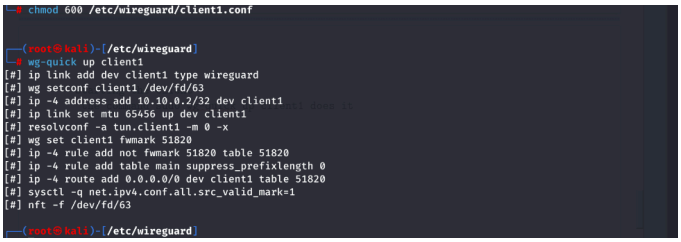
[Peer]
PublicKey = <server_publickey>
Endpoint = <SERVER_PUBLIC_IP>:51820
AllowedIPs = 0.0.0.0/0
PersistentKeepalive = 25

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



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 ion our case are[OpenVPN, Easy-SRA, Kernel, Networking tools] are in the latest stable versions.

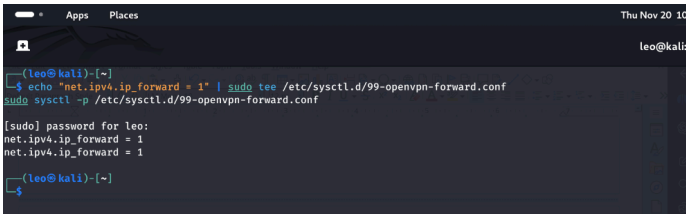
Enabling IP forwarding.

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.



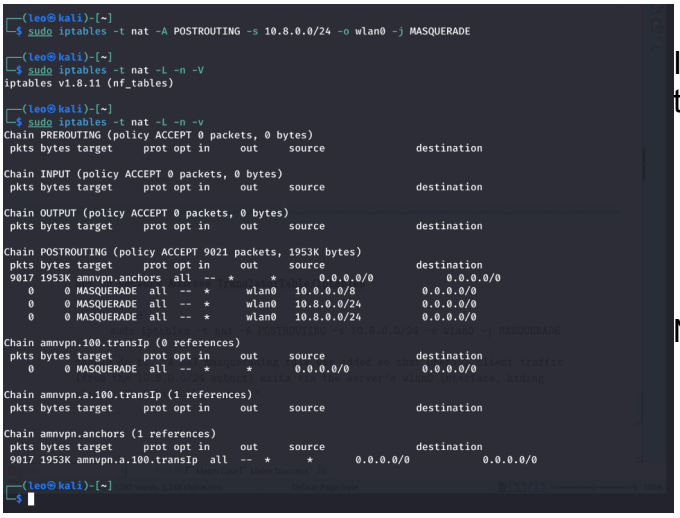
the expected output on tunning the command: **sysctl -p …** should be net.ipv4.ip_forward = 1.

Set up Network Address TranslatorTable/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 IP tables and make them persistent use the command.

sudo apt install iptables-persistent

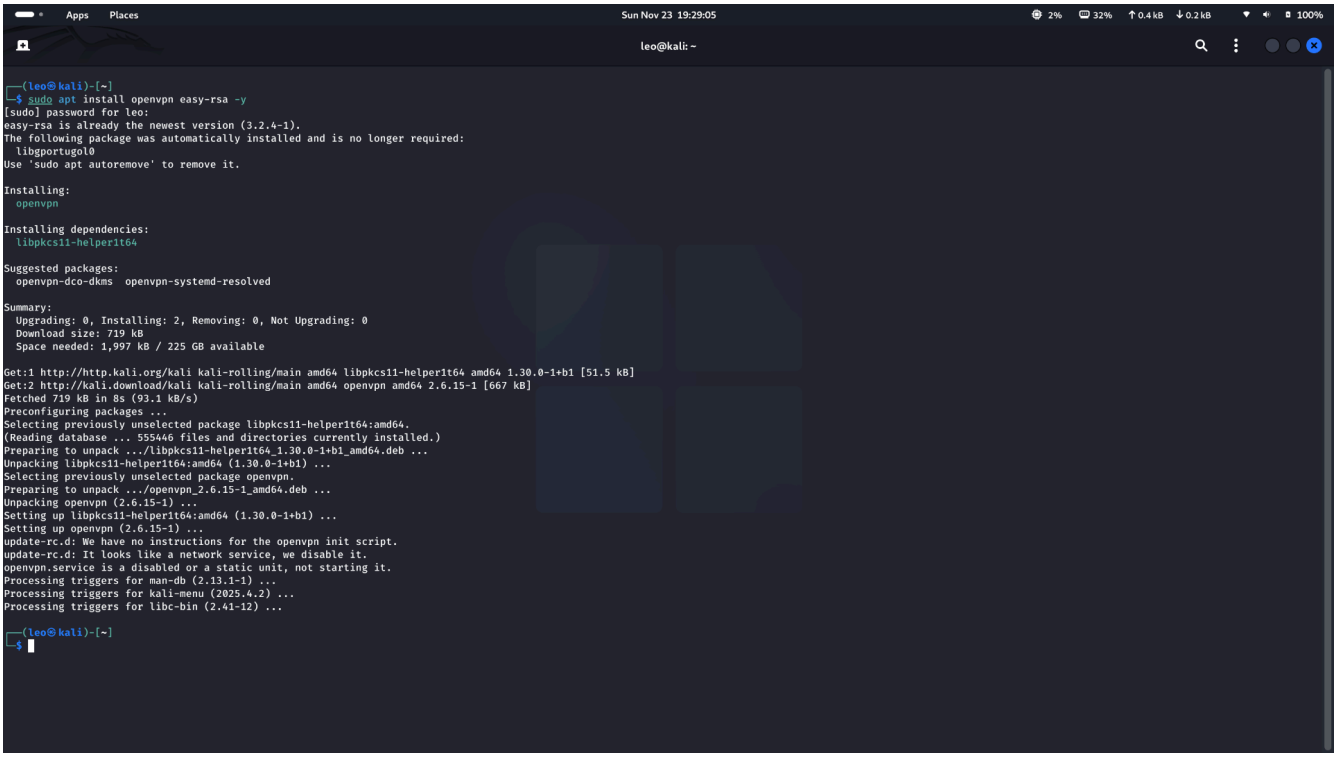
sudo netfilter-persistent save

this ensures that the NAT rule persistent across reboot.

Install OpenVPN and Easy-RSA

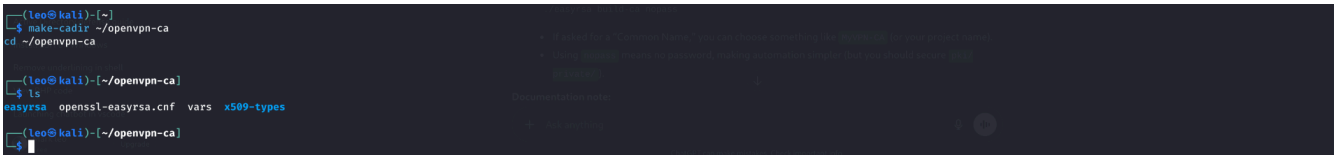
to install OpenVpn run the command

sudo apt install openvpn easy-rsa -y


Apps
Places
Sun Nov 23 19:29:05
2%
32%
0.4 kB
0.2 kB
100%
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:
libgportugol0
Use 'sudo apt autoremove' to remove it.
Installing:
openvpn
Installing dependencies:
libpkcs11-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 libpkcs11-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 0s (93.1 kB/s)
Preconfiguring packages ...
Selecting previously unselected package libpkcs11-helper1t64:amd64.
(Reading database ... 555446 files and directories currently installed.)
Preparing to unpack .../libpkcs11-helper1t64_1.30.0-1+b1_amd64.deb ...
Unpacking libpkcs11-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 libpkcs11-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:



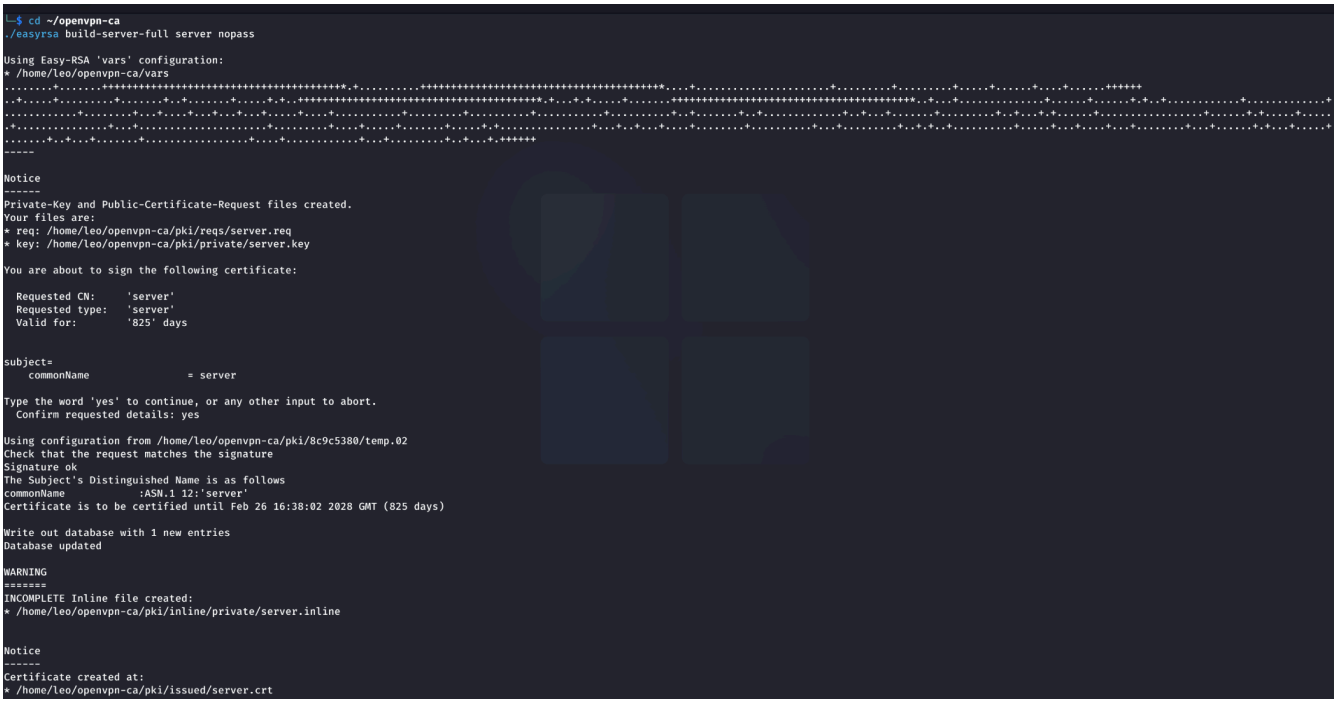
the commands above creates the directory and navigates to it.

2. Initialize the PKI

3. Build a Certificate Authority

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.

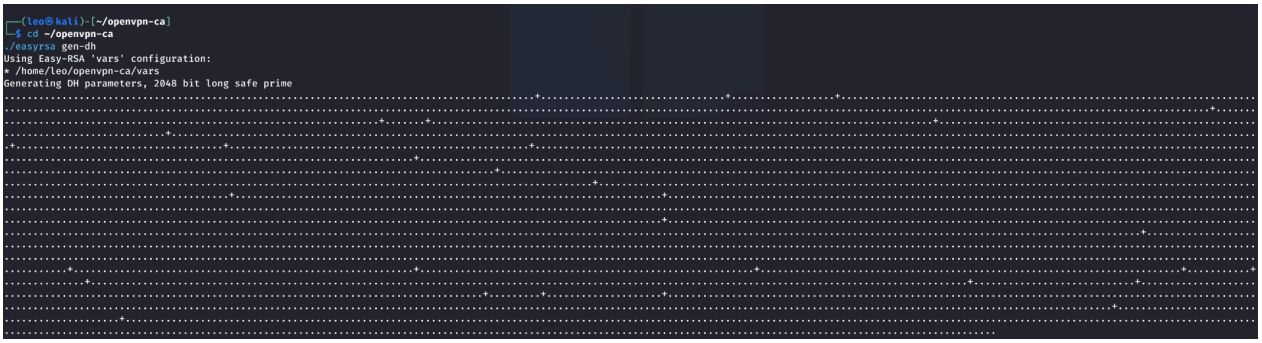


server is the name of the certificate.

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



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

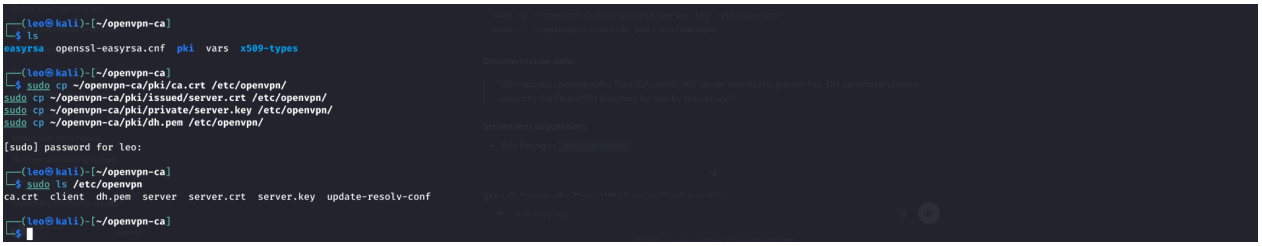
Copy ther materials to the OpenVPN directory in the /etc directory with the commands

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 

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.

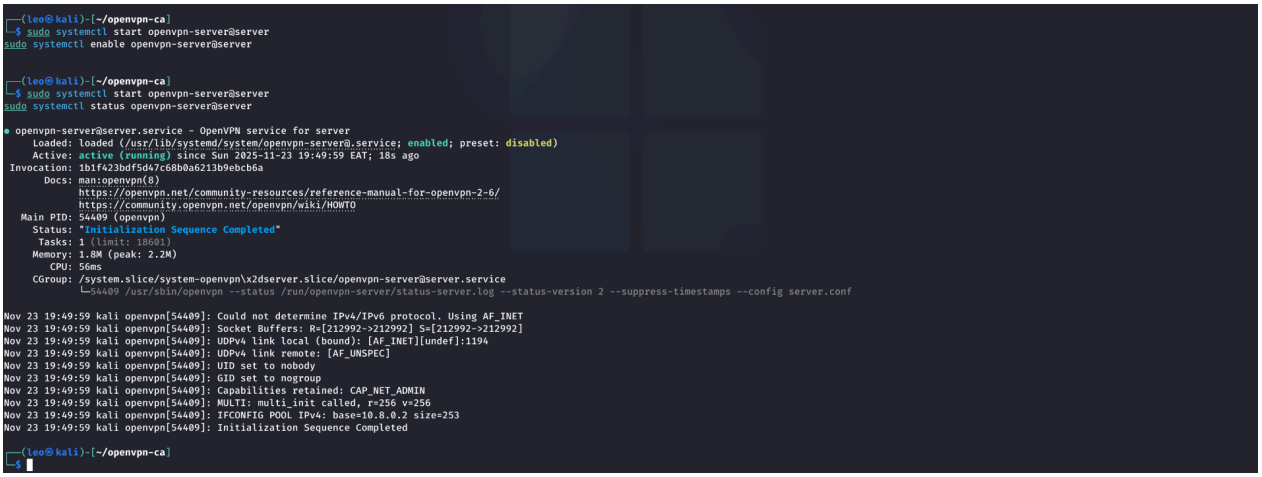
In the openvpn directory create a directory named server and a file named server.conf as per the image below.

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



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

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