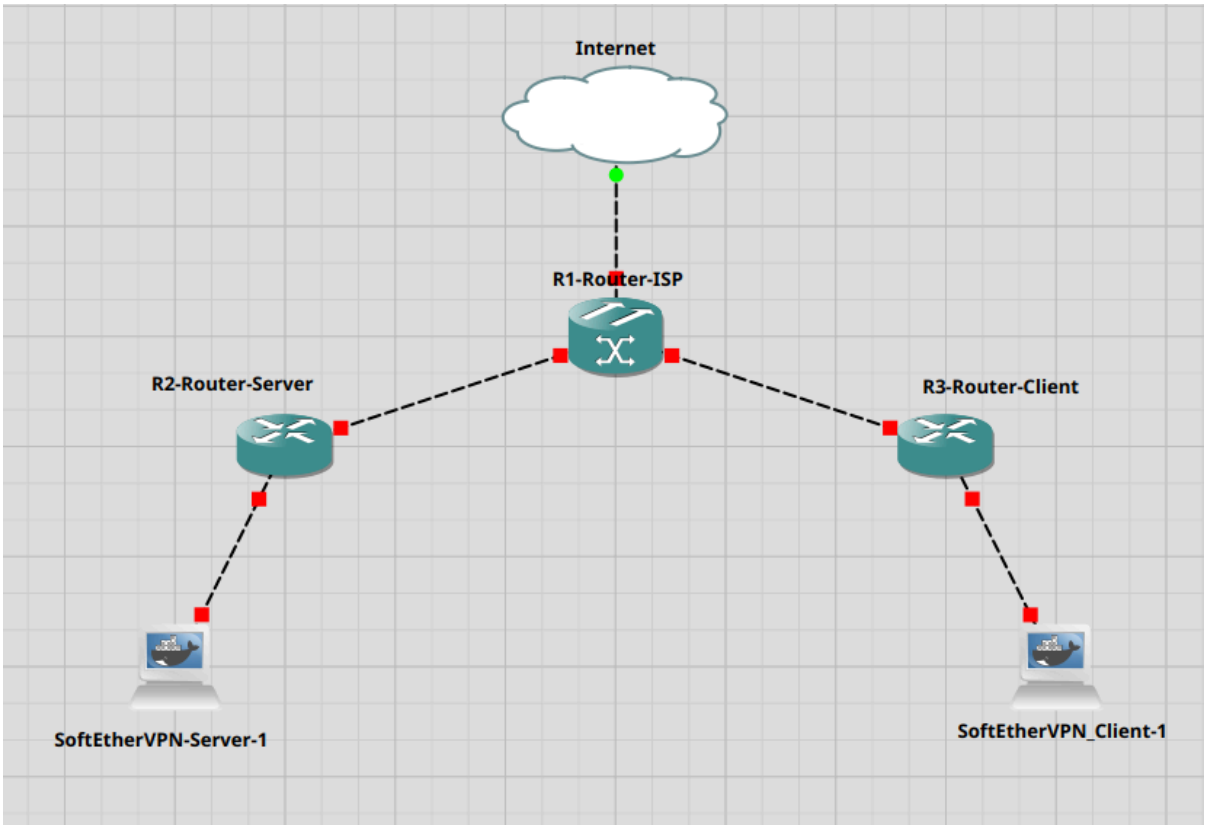
GNS3 Public IP, VPN Across the Internet

▼ Topology Definition




```
                [Internet Cloud]
                       |
[Server] ↔ [Router 2] ↔ [ISP Router] ↔ [Router 3] ↔ [Client]
```

IP Addressing

Device	Interface	IP Address	Description
Router 2	LAN (Fa0/0)	10.0.1.1/24	Private Network 1

Router 2	WAN (Fa0/1)	203.0.113.1/24	Public IP (RFC 5737)
ISP Router	Interface Fa0/0	203.0.113.254/24	Connected to Router 2
ISP Router	Interface Fa0/1	198.51.100.254/24	Connected to Router 3
ISP Router	Interface Fa1/0	Internet Cloud GNS3	
Router 3	WAN (Fa0/1)	198.51.100.1/24	Public IP (RFC 5737)
Router 3	LAN (Fa0/0)	10.0.2.1/24	Private Network 2
Server	eth0	10.0.1.2/24	Private IP
Client	eth0	10.0.2.2/24	Private IP

▼ Technologies Involved and GNS3 configuration

We have these devices in the confiugration:

- **2x Edge Routers**
- **1x ISP Router**
- **2x Docker containers**
- **1x Cloud Node**

Routers

The three routers are all CISCO routers, in particular we have downloaded the image of a Cisco 7200 124-24.T5 router, that we can easily import in out GNS3 project.

To import the router in our project we use this process:

- Press on the left on the router icon
- Press on the bottom on the “new Template” section
- now go next on “Install an appliance from the GNS3 server”
- In the filters section write CISCO
- go down until you find CISCO 7200
- press install and then next to install on the local device

- now go next again and find the right version of the router, that in this case is the “124-24.T5” one.
- We press on it and we press “import”, and then we go in the folder where there is our router, so the one that i have attached to the project. Now we have a new router in our project.

Containers

In this case, we are creating two main containers, these are their images:

- Server: siomiz/softethervpn:latest
- Client: ubuntu:latest

This because, the server needs to be a server that runs the softetherVPN module, instead the Client might be a general purpose machine that would like to connect to a VPN server.

To add these containers, this is the process:

- we open the GNS3 project
- we go in the “Edit” section in the navbar
- then we press on “Preferences”.
- At this point we are in the Preferences section of our project where we are able to manage all its features.
- In particular in this case we need the Docker containers section, where we are able to add new containers.
    - we press on “new” to add a new container in the project, and we write the name of the image we would like, so in the server case we use “siomiz/softethervpn:latest”, in the client case we insert “ubuntu:latest”.
    - now we choose the name and we go next until finished.

▼ Step 1: Build the Topology in GNS3

1. Add these devices:
    - 2x Edge Routers (Router 1 and Router 2 – Cisco IOSv recommended)

- 1x ISP Router (Cisco IOSv or VyOS)
- 2x Docker containers (SoftEther Server/Client)
- 1x Cloud Node (To bridge the traffic to internet)

2. Connect them:

```
# Connect the server container to the router of its subnet
Server ↔ Router 2 (Fa0/0)

# Connect the router of the server's subnet to the one of the ISP
Router 2 (Fa0/1) ↔ ISP Router (Fa0/0)

# Connect the ISP router to the router of the Client's Subnet
ISP Router (Fa0/1) ↔ Router 3 (Fa0/1)

# Connect the ISP router to the internet cloud (choose the right interfac
e)
ISP Router (Fa1/0) ↔ Cloud Node

# Connect the Client container to the router of its subnet
Router 3 (Fa0/0) ↔ Client
```

3. Configure the Client and the Server.
    - Server
        - Open the configurations of the server container.
        - Go in the advanced settings
        - Add these two routes in the additional directories section

        ```
        /server
        /usr/vpnserver
        ```

        now, there will be created two addictional directories in the folder of the server container in the project folder. In this way, we could add the files needed for the configuration in a static way inside the container.

- Client
  - Open the configurations of the server container.
  - Go in the advanced settings
  - Add these two routes in the additional directories section

```
/client
```

At the same way now we are able to give the needed file to the client container by means of the new volume created.

▼ Step 2: Configure the ISP Router

2.1 R1-Router-ISP

```
enable
configure terminal
```

Configure the interface to the Router 1, the one connected to the server, and we assign a random public IP address. (This address is also used in the configuration of the IPsec and TLS modules, if they are changed here this means that they need to be changed also in the configuration files of the two protocols)

```
interface FastEthernet0/0
 ip address 203.0.113.254 255.255.255.0
 no shutdown
```

Configure the interface to the Router 2, the one connected to the client, and also here we assign a random IP address.

```
exit

interface FastEthernet0/1
```

```
 ip address 198.51.100.254 255.255.255.0
 no shutdown
```

Now we need to configure the interface to the Internet. We need to connect this interface of the ISP router to the interface of the host machine that is actually being used to access the internet (Wireless one or Ethernet one). In this case, we cannot manually assign the IP address to the interface, but instead we need to exploit the default DHCP server, asking it to assign us a new IP address.

In the Politecnico's default DHCP server, or in general, if your interface does not has got a MAC address, it will be filtered and the request will not be answered. This means that we need to assign to the interface a MAC address that could be a random one or the one of the host machine.

To list the MAC address of the interested interface on the host machine, you can run on the Linux terminal the current command:

```
ip addr show <interface-name>
```

Then the mac address will be listed after the word "link/ether". We can now assign that MAC address to the interface of the ISP router in the GNS3 Project.

```
exit

interface FastEthernet1/0
 mac-address 1cce.513f.9820 <spoofed-mac-host-machine> (format is: 1111.2
 ip address dhcp
 no shutdown
```

exit from the interface configuration

```
exit
```

Here now we can define which are the routes for the packets destinated to the Server's subnet, and to the Client's Subnet

```
ip route 198.51.100.0 255.255.255.0 FastEthernet0/1
ip route 203.0.113.0 255.255.255.0 FastEthernet0/0
```

We define now the default route, the one where all the data will be sent if not related to the client server communication.

```
ip route 0.0.0.0 0.0.0.0 FastEthernet1/0
```

Now we can end the configuration and write the changes in the memory.

```
end
wr
```

▼ Step 3: Configure Edge Routers

3.1 R2-Router-Server

```
enable
configure terminal
```

Configure the interface connected to the server host

```
interface FastEthernet0/0
  ip address 10.0.1.1 255.255.255.0
  ip nat inside
  no shutdown
```

Configure the interface connected to the ISP router

```
exit

interface FastEthernet0/1
  ip address 203.0.113.1 255.255.255.0
```

```
ip nat outside
no shutdown
```

We now need to configure the NAT. In particular we are mapping the services offered by the server to the same ports of the router. These ports are the ones needed by the server to server the IPsec service and the TLS one. (Port 500 is used for the ISAKMP connection, Port 4500 is used for the ESP tunnel, Port 443 is used by the TLS protocol)

```
exit

ip nat inside source static udp 10.0.1.2 500 203.0.113.1 500
ip nat inside source static udp 10.0.1.2 4500 203.0.113.1 4500
ip nat inside source static tcp 10.0.1.2 443 203.0.113.1 443
```

This next command defines which local (LAN) IP addresses are allowed to be translated by NAT, in particular we are allowing the traffic as there were more hosts in the subnet.

```
access-list 1 permit 10.0.1.0 0.0.0.255
```

With this other command, we are translating the source IP of outbound packets from the 10.0.1.0/24 network to the public IP of FastEthernet0/1, using PAT.

```
ip nat inside source list 1 interface FastEthernet0/1 overload
```

Define now which is the default gateway for this router, so which is the next hop, that will be the ISP router’s interface

```
ip route 0.0.0.0 0.0.0.0 FastEthernet0/1
```

now we can end the configuration and write everything in the memory of the router.

```
end
```

```
wr
```

3.2 R3-Router-Client

```
enable
configure terminal
```

Configure the interface connected to the client host

```
interface FastEthernet0/0
  ip address 10.0.2.1 255.255.255.0
  ip nat inside
  no shutdown
```

Configure the interface connected to the ISP Router

```
exit

interface FastEthernet0/1
  ip address 198.51.100.1 255.255.255.0
  ip nat outside
  no shutdown
```

Define now which is the default gateway for this router, so which is the next hop, that will be the ISP router’s interface

```
exit

ip route 0.0.0.0 0.0.0.0 FastEthernet0/1
```

This next command defines which local (LAN) IP addresses are allowed to be translated by NAT, in particular we are allowing the traffic as there were more hosts in the subnet.

```
access-list 1 permit 10.0.2.0 0.0.0.255
```

With this other command, we are translating the source IP of outbound packets from the 10.0.2.0/24 network to the public IP of FastEthernet0/1, using PAT.

```
ip nat inside source list 1 interface FastEthernet0/1 overload
```

we can now end the configuration and write everything in the router’s memory

```
end
wr
```

▼ Step 4: Configure SoftEtherVPN Server & Client

4.1 Server Configuration (10.0.1.2)

Configure the IP address of the of the Server Container on its terminal.

```
ip addr add 10.0.1.2/24 dev eth0 &&
ip route add default via 10.0.1.1
```

check if the server is listening on the right ports

```
ss -tuln
```

4.2 Client Configuration (10.0.2.2)

```
ip addr add 10.0.2.2/24 dev eth0 &&
ip route add default via 10.0.2.1 &&
```

```
cp /client/ipsec.conf /etc/ &&
cp /client/ipsec.secrets /etc
```

▼ Step 5: Test Connectivity

5.1 Basic Reachability

```
# From Router 1:
ping 198.51.100.254  # Should reach ISP Router
ping 198.51.100.1     # Should reach Router 2

# From Router 2:
ping 203.0.113.254    # Should reach ISP Router
ping 203.0.113.1      # Should reach Router 1

#From Client:
ping 203.0.113.1

#From Server:
ping 198.51.100.1
```

▼ VPN Configuration

▼ Server Files

In the case of the server side, only one file of the SoftEtherVPN module will be eoungh to let the client connect to the VPN both by means of the IPsec module and with the TLS one.

vpn_server.config:

```
# Software Configuration File
# ---------------------------
#
# You may edit this file when the VPN Server / Client / Bridge program is n
#
```

```
# In prior to edit this file manually by your text editor,
# shutdown the VPN Server / Client / Bridge background service.
# Otherwise, all changes will be lost.
#
declare root
{
   uint ConfigRevision 20
   bool IPsecMessageDisplayed false
   string Region $

   declare DDnsClient
   {
      bool Disabled false
      byte Key LKaxqKfenGwl+tbuXwaYYrMObYE=
      string LocalHostname 349e4e13440e
      string ProxyHostName $
      uint ProxyPort 0
      uint ProxyType 0
      string ProxyUsername $
   }
   declare IPsec
   {
      bool EtherIP_IPsec true
      string IPsec_Secret ciao
      string L2TP_DefaultHub DEFAULT
      bool L2TP_IPsec false
      bool L2TP_Raw false

      declare EtherIP_IDSettingsList
      {
      }
   }
   declare ListenerList
   {
      declare Listener0
      {
```

```
        bool DisableDos false
        bool Enabled true
        uint Port 443
    }
    declare Listener1
    {
        bool DisableDos false
        bool Enabled true
        uint Port 500
    }
    declare Listener2
    {
        bool DisableDos false
        bool Enabled true
        uint Port 992
    }
    declare Listener3
    {
        bool DisableDos false
        bool Enabled true
        uint Port 4500
    }
    declare Listener4
    {
        bool DisableDos false
        bool Enabled true
        uint Port 5555
    }
}
declare LocalBridgeList
{
    bool DoNotDisableOffloading false
}
declare ServerConfiguration
{
    bool AcceptOnlyTls false
```

```
        uint64 AutoDeleteCheckDiskFreeSpaceMin 104857600
        uint AutoDeleteCheckIntervalSecs 300
        uint AutoSaveConfigSpan 300
        bool BackupConfigOnlyWhenModified true
        string CipherName DHE-RSA-AES256-SHA
        uint CurrentBuild 9799
        bool DisableCoreDumpOnUnix false
        bool DisableDeadLockCheck false
        bool DisableDosProction false
        bool DisableGetHostNameWhenAcceptTcp false
        bool DisableIntelAesAcceleration false
        bool DisableIPsecAggressiveMode false
        bool DisableIPv6Listener false
        bool DisableJsonRpcWebApi false
        bool DisableNatTraversal false
        bool DisableOpenVPNServer false
        bool DisableSessionReconnect false
        bool DisableSSTPServer false
        bool DontBackupConfig false
        bool EnableVpnAzure false
        bool EnableVpnOverDns false
        bool EnableVpnOverIcmp false
        byte HashedPassword 57GljujThI2IEV43eoHMiT7X3Ek=
        string KeepConnectHost keepalive.softether.org
        uint KeepConnectInterval 50
        uint KeepConnectPort 80
        uint KeepConnectProtocol 1
        uint64 LoggerMaxLogSize 1073741823
        uint MaxConcurrentDnsClientThreads 512
        uint MaxConnectionsPerIP 256
        uint MaxUnestablishedConnections 1000
        bool NoHighPriorityProcess false
        bool NoLinuxArpFilter false
        bool NoSendSignature false
        string OpenVPNDefaultClientOption dev-type$20tun,link-mtu$201500
        string OpenVPN_UdpPortList 1194
```

```
    bool SaveDebugLog true
    byte ServerCert MIIDpjCCAo6gAwIBAgIBADANBgkqhkiG9w0BAQsFAI
    byte ServerKey MIIEpQIBAAKCAQEAwWwygD+PDERQtoUtijdf8E5nM
    uint ServerLogSwitchType 4
    uint ServerType 0
    bool StrictSyslogDatetimeFormat false
    bool Tls_Disable1_0 false
    bool Tls_Disable1_1 false
    bool Tls_Disable1_2 false
    bool Tls_Disable1_3 false
    bool UseKeepConnect true
    bool UseWebTimePage false
    bool UseWebUI false

    declare GlobalParams
    {
        uint FIFO_BUDGET 10240000
        uint HUB_ARP_SEND_INTERVAL 5000
        uint IP_TABLE_EXPIRE_TIME 60000
        uint IP_TABLE_EXPIRE_TIME_DHCP 300000
        uint MAC_TABLE_EXPIRE_TIME 600000
        uint MAX_BUFFERING_PACKET_SIZE 2560000
        uint MAX_HUB_LINKS 1024
        uint MAX_IP_TABLES 65536
        uint MAX_MAC_TABLES 65536
        uint MAX_SEND_SOCKET_QUEUE_NUM 128
        uint MAX_SEND_SOCKET_QUEUE_SIZE 2560000
        uint MAX_STORED_QUEUE_NUM 1024
        uint MEM_FIFO_REALLOC_MEM_SIZE 655360
        uint MIN_SEND_SOCKET_QUEUE_SIZE 320000
        uint QUEUE_BUDGET 2048
        uint SELECT_TIME 256
        uint SELECT_TIME_FOR_NAT 30
        uint STORM_CHECK_SPAN 500
        uint STORM_DISCARD_VALUE_END 1024
        uint STORM_DISCARD_VALUE_START 3
```

```
    }
    declare ServerTraffic
    {
      declare RecvTraffic
      {
        uint64 BroadcastBytes 640744
        uint64 BroadcastCount 10504
        uint64 UnicastBytes 218358
        uint64 UnicastCount 5199
      }
      declare SendTraffic
      {
        uint64 BroadcastBytes 0
        uint64 BroadcastCount 0
        uint64 UnicastBytes 218358
        uint64 UnicastCount 5199
      }
    }
    declare SyslogSettings
    {
      string HostName $
      uint Port 0
      uint SaveType 0
    }
  }
  declare VirtualHUB
  {
    declare DEFAULT
    {
      uint64 CreatedTime 1744089202177
      byte HashedPassword CGFhwOK7TZMhrLeE3wQ83csfFKo=
      uint64 LastCommTime 1744528732594
      uint64 LastLoginTime 1744089202176
      uint NumLogin 0
      bool Online true
      bool RadiusConvertAllMsChapv2AuthRequestToEap false
```

```
        string RadiusRealm $
        uint RadiusRetryInterval 0
        uint RadiusServerPort 1812
        string RadiusSuffixFilter $
        bool RadiusUsePeapInsteadOfEap false
        byte SecurePassword FRzHI15z2uczdYHXcBe8ubgcg1g=
        uint Type 0

        declare AccessList
        {
        }
        declare AdminOption
        {
            uint allow_hub_admin_change_option 0
            uint deny_bridge 0
            uint deny_change_user_password 0
            uint deny_empty_password 0
            uint deny_hub_admin_change_ext_option 0
            uint deny_qos 0
            uint deny_routing 0
            uint max_accesslists 0
            uint max_bitrates_download 0
            uint max_bitrates_upload 0
            uint max_groups 0
            uint max_multilogins_per_user 0
            uint max_sessions 0
            uint max_sessions_bridge 0
            uint max_sessions_client 0
            uint max_sessions_client_bridge_apply 0
            uint max_users 0
            uint no_access_list_include_file 0
            uint no_cascade 0
            uint no_change_access_control_list 0
            uint no_change_access_list 0
            uint no_change_admin_password 0
            uint no_change_cert_list 0
```

```
        uint no_change_crl_list 0
        uint no_change_groups 0
        uint no_change_log_config 0
        uint no_change_log_switch_type 0
        uint no_change_msg 0
        uint no_change_users 0
        uint no_delay_jitter_packet_loss 0
        uint no_delete_iptable 0
        uint no_delete_mactable 0
        uint no_disconnect_session 0
        uint no_enum_session 0
        uint no_offline 0
        uint no_online 0
        uint no_query_session 0
        uint no_read_log_file 0
        uint no_securenat 0
        uint no_securenat_enabledhcp 0
        uint no_securenat_enablenat 0
    }
    declare CascadeList
    {
    }
    declare LogSetting
    {
        uint PacketLogSwitchType 4
        uint PACKET_LOG_ARP 0
        uint PACKET_LOG_DHCP 1
        uint PACKET_LOG_ETHERNET 0
        uint PACKET_LOG_ICMP 0
        uint PACKET_LOG_IP 0
        uint PACKET_LOG_TCP 0
        uint PACKET_LOG_TCP_CONN 1
        uint PACKET_LOG_UDP 0
        bool SavePacketLog false
        bool SaveSecurityLog false
        uint SecurityLogSwitchType 4
```

```
}
declare Message
{
}
declare Option
{
   uint AccessListIncludeFileCacheLifetime 30
   uint AdjustTcpMssValue 0
   bool ApplyIPv4AccessListOnArpPacket false
   bool AssignVLanIdByRadiusAttribute false
   bool BroadcastLimiterStrictMode false
   uint BroadcastStormDetectionThreshold 0
   uint ClientMinimumRequiredBuild 0
   bool DenyAllRadiusLoginWithNoVlanAssign false
   uint DetectDormantSessionInterval 0
   bool DisableAdjustTcpMss false
   bool DisableCheckMacOnLocalBridge false
   bool DisableCorrectIpOffloadChecksum false
   bool DisableHttpParsing false
   bool DisableIPParsing false
   bool DisableIpRawModeSecureNAT false
   bool DisableKernelModeSecureNAT false
   bool DisableUdpAcceleration false
   bool DisableUdpFilterForLocalBridgeNic false
   bool DisableUserModeSecureNAT false
   bool DoNotSaveHeavySecurityLogs false
   bool DropArpInPrivacyFilterMode true
   bool DropBroadcastsInPrivacyFilterMode true
   bool FilterBPDU false
   bool FilterIPv4 false
   bool FilterIPv6 false
   bool FilterNonIP false
   bool FilterOSPF false
   bool FilterPPPoE false
   uint FloodingSendQueueBufferQuota 33554432
   bool ManageOnlyLocalUnicastIPv6 true
```

```
    bool ManageOnlyPrivateIP true
    uint MaxLoggedPacketsPerMinute 0
    uint MaxSession 0
    bool NoArpPolling false
    bool NoDhcpPacketLogOutsideHub true
    bool NoEnum false
    bool NoIpTable false
    bool NoIPv4PacketLog false
    bool NoIPv6AddrPolling false
    bool NoIPv6DefaultRouterInRAWhenIPv6 true
    bool NoIPv6PacketLog false
    bool NoLookBPDUBridgeId false
    bool NoMacAddressLog true
    bool NoManageVlanId false
    bool NoPhysicalIPOnPacketLog false
    bool NoSpinLockForPacketDelay false
    bool RemoveDefGwOnDhcpForLocalhost true
    uint RequiredClientId 0
    uint SecureNAT_MaxDnsSessionsPerIp 0
    uint SecureNAT_MaxIcmpSessionsPerIp 0
    uint SecureNAT_MaxTcpSessionsPerIp 0
    uint SecureNAT_MaxTcpSynSentPerIp 0
    uint SecureNAT_MaxUdpSessionsPerIp 0
    bool SecureNAT_RandomizeAssignIp false
    bool SuppressClientUpdateNotification false
    bool UseHubNameAsDhcpUserClassOption false
    bool UseHubNameAsRadiusNasId false
    string VlanTypeId 0x8100
    bool YieldAfterStorePacket false
}
declare SecureNAT
{
    bool Disabled false
    bool SaveLog false

    declare VirtualDhcpServer
```

```
        {
            string DhcpDnsServerAddress 192.168.30.1
            string DhcpDnsServerAddress2 0.0.0.0
            string DhcpDomainName $
            bool DhcpEnabled true
            uint DhcpExpireTimeSpan 7200
            string DhcpGatewayAddress 192.168.30.1
            string DhcpLeaseIPEnd 192.168.30.200
            string DhcpLeaseIPStart 192.168.30.10
            string DhcpPushRoutes $
            string DhcpSubnetMask 255.255.255.0
        }
        declare VirtualHost
        {
            string VirtualHostIp 192.168.30.1
            string VirtualHostIpSubnetMask 255.255.255.0
            string VirtualHostMacAddress 5E-C0-9A-6B-CA-0C
        }
        declare VirtualRouter
        {
            bool NatEnabled true
            uint NatMtu 1500
            uint NatTcpTimeout 3600
            uint NatUdpTimeout 1800
        }
    }
    declare SecurityAccountDatabase
    {
        declare CertList
        {
        }
        declare CrlList
        {
        }
        declare GroupList
        {
```

```
        }
        declare IPAccessControlList
        {
        }
        declare UserList
        {
           declare user1
           {
              byte AuthNtLmSecureHash xh42jVOL34s+AqTYKN8b0g==
              byte AuthPassword ObNWU1DckHL0Xg4HuyRAMKilANY=
              uint AuthType 1
              uint64 CreatedTime 1744089204585
              uint64 ExpireTime 0
              uint64 LastLoginTime 0
              string Note $
              uint NumLogin 0
              string RealName $
              uint64 UpdatedTime 1744089204799

              declare Traffic
              {
                 declare RecvTraffic
                 {
                    uint64 BroadcastBytes 0
                    uint64 BroadcastCount 0
                    uint64 UnicastBytes 0
                    uint64 UnicastCount 0
                 }
                 declare SendTraffic
                 {
                    uint64 BroadcastBytes 0
                    uint64 BroadcastCount 0
                    uint64 UnicastBytes 0
                    uint64 UnicastCount 0
                 }
              }
```

```
            }
          }
        }
        declare Traffic
        {
          declare RecvTraffic
          {
            uint64 BroadcastBytes 640744
            uint64 BroadcastCount 10504
            uint64 UnicastBytes 218358
            uint64 UnicastCount 5199
          }
          declare SendTraffic
          {
            uint64 BroadcastBytes 0
            uint64 BroadcastCount 0
            uint64 UnicastBytes 218358
            uint64 UnicastCount 5199
          }
        }
      }
    }
    declare VirtualLayer3SwitchList
    {
    }
}
```

▼ Client Files

▼ IPSec

on the client side, run this command to see the whole debug of the IPSec strongswan module

```
ipsec start --nofork --debug-all
```

5.2 VPN Tunnel Test

```
# On Client container:
ipsec restart
ipsec up myserver
ipsec status  # Should show "INSTALLED, TUNNEL"
```

Files

ipsec.conf file:

```
config setup
    charondebug="ike 2, knl 2, cfg 2"   # Debugging levels for IKE, kerne
    uniqueids=yes                  # Ensure unique IDs for connections.

conn softether
    # Local settings
    left=%any
    leftsubnet=10.0.2.0/24           # Subnet to encrypt (mandatory)
    leftid=@client                 # Unique client ID
    # Remote settings
    right=203.0.113.1
    rightid=10.0.1.2               # Match the server's actual ID
    rightsubnet=0.0.0.0/0            # All traffic via VPN
    # NAT-Traversal
    forceencaps=yes                # Force UDP encapsulation
    # Phase 1 (IKEv1)
    keyexchange=ikev1
    ike=aes256-sha1-modp2048!         # Encryption: AES-256, SHA1, DH
    esp=aes128-sha1-modp1024!         # ESP encryption: AES-128, SHA
    aggressive=no                  # Disable unless server requires it
    # Authentication
    leftauth=psk
    rightauth=psk
    dpdaction=restart               # Restart connection on dead peer dete
    dpddelay=30s                   # Delay between DPD messages.
```

```
    dpdtimeout=120s                 # Timeout for DPD.
    auto=start
```

ipsec.secrets file:

This file is the one needed for the management of the shared secret between the client and the server.

```
# IPsec secrets configuration file.
# This file contains the pre-shared key (PSK) for the IPsec connection.

%any 203.0.113.1 : PSK "ciao"  # Client IP, Server IP, and the shared PSK
```

▼ TLS / SSL

DEMO

Now, in order to run the VPN with the TLS module, we will run this command basically. This command allows us to open the VPN in background by means of the & at the end of the command, so that we can keep using the terminal as we need.

```
cd client
openvpn --config softether.ovpn &
```

We could now see opening a wireshark instance on a wire of between the two hosts, that there is an SSL tunnel between the two.

credentials.txt:

These credentials are the ones of the user inside the server.

```
user1
ciao
```

softether.ovpn:

We have now the configuration file of the openvpn vpn to connect to the remote server.

```
client
dev tun
proto tcp
remote 203.0.113.1 443
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
remote-cert-tls server
verify-x509-name da3af5075c51 name
auth-user-pass credentials.txt
cipher AES-128-CBC
data-ciphers AES-128-CBC
mssfix 1450
verb 4
```

ca.crt

Now we have the certificate of the server, so that we can perform the authentication of the server with its cert. This is just a random certificate so we put it here in plaintext.

```
-----BEGIN CERTIFICATE-----
MIIDpjCCAo6gAwIBAgIBADANBgkqhkiG9w0BAQsFADBSMRUwEwYDV
NzVjNTExFTATBgNVBAoMDGRhM2FmNTA3NWM1MTEVMBMGA1UECv
MQswCQYDVQQGEwJVUzAeFw0yNTA0MDgxNDEzMjJaFw0zNzEyMz
BgNVBAMMDGRhM2FmNTA3NWM1MTEVMBMGA1UECgwMZGEzYWY
DAxkYTNhZjUwNzVjNTExCzAJBgNVBAYTAlVTMIIBIjANBgkqhkiG9w0E
MIIBCgKCAQEAwWwygD+PDERQtoUtijdf8E5nMCiZXVnj3zyir18PcmDtD
QBmDTFToxVvjRFbMSagu836AZp3H4+IheXPKSm5L2+XAioUOkbt9+jJ
/3QJRuGescaZvpmfRVrZJ7o7yNV7HZKZQpAyUk6vw+WJw/qPmN4Pbs
6t7ugM+2bQYPEnlB1+NyEm1WieF8Y0xPMzIL9eFiynzENADS/646FIHA1
DGsgKuTcta6ptb/SU/OF/ozKUmKqyBhChJF0NItgKn5g15xgly72djIz56G
xzyAQPHl0cJO+QIDAQABo4GGMIGDMA8GA1UdEwEB/wQFMAMBAf8w
MGMGA1UdJQRcMFoGCCsGAQUFBwMBBggrBgEFBQcDAgYIKwYBBQ
```

```
BggrBgEFBQcDBQYIKwYBBQUHAwYGCCsGAQUFBwMHBggrBgEFBQc
DQYJKoZIhvcNAQELBQADggEBALELFGxRsrbxY3T5xXPL8FVFBx2yqoT
bYASeZa5vCF7cWyMtmxt+KDZobyI98ZQvBvmPljUiIOKhvUuiokvtpvFZp
lbq6AwwXFyqTiksDKQsyEi/LjhQKone6thNCnPhlecNM+oxCzFfI2Ndt5ry
fwgcz98CJFCSNbxZknsNsFhaR6gTQM1rNVOO0+C5uanmOJN9UbT6U
TxFS1SIcoLk66Mm/wpyWltbRKwkGqDfK7Alcd4xYgvL4BOcA+D9kdFU/I
w3UVzAYxeIfrm2ghFQRSCrIvvt8=
-----END CERTIFICATE-----
EOF
```