SoftEther VPN Lab Activity

Virtual Private Networks Across the Internet

Renato Mignone

Student ID: s336973

MSc in Cybersecurity Engineering

Politecnico di Torino

Network and Cloud Security Practical Activity

License

Acknowledgments

The author would like to thank Professor **Daniele Bringhenti**, Professor **Fulvio Valenza**, Dr. **Gianmarco Bachiorrini** and the Politecnico di Torino for providing the foundational knowledge that made this work possible.

Contents

1 Introduction

This Project involves the implementation and analysis of **Virtual Private Networks** (VPNs) using **SoftEther VPN** technology across an Internet infrastructure. VPNs are fundamental network security technologies that provide secure connectivity across shared, potentially untrusted networks by creating encrypted tunnels that preserve the confidentiality, integrity, and authenticity of data transmission.

The objective of this lab is to demonstrate how modern VPN technologies can establish secure communication channels between geographically separated networks, mimicking real-world scenarios in which branch offices or remote users need secure access to corporate resources. Through practical implementation, we will examine two different VPN protocols: IPSec and TLS/SSL, comparing their different approaches and methods to achieve network security.

1.1 Laboratory Objectives

The main objectives of this lab activity are:

- **VPN Basics:** Learn the essentials of Virtual Private Networks and their application in modern network security architectures.
- **SoftEther VPN Deployment:** Install and deploy SoftEther VPN server to utilize multi-protocol VPN services, as IPSec and TLS-based connections.
- **Network Topology Design:** Design a realistic network topology using GNS3 that simulates Internet interconnectivity between two remote sites over an ISP infrastructure.
- **Protocol Comparison:** Contrast and compare different VPN protocols (IPSec vs. TLS) in terms of security features and complexity.
- **Traffic Analysis:** Perform packet capture with Wireshark to observe how differently the VPN protocols encapsulate and protect network traffic differently.
- **Security Assessment:** Examine the security mechanisms provided by each VPN implementation, including encryption, authentication, and integrity protection.

1.2 Laboratory Environment and Network Topology

This lab simulates a scenario in which two geographically separated private networks need to commmunicate securely over the Internet. The topology consists of:

- **Server Site:** A private network (10.0.1.0/24) hosting the SoftEther VPN server
- **Client Site:** A remote private network (10.0.2.0/24) with VPN client capability
- **Internet Infrastructure:** Simulated ISP network providing connectivity between the sites
- **VPN Gateways:** Edge routers providing NAT and routing capabilities, to which a public IP address is assigned

The configuration demonstrates how VPN technology facilitates secure site-to-site connections, allowing hosts whithin different private networks to communicate as if connected to the same local network, while preserving security attributes across the untrusted Internet infrastructure.

1.3 System Requirements

In order to successfully complete this lab activity, the following system requirements must be met:

System Requirements

Operating System:

- Linux distribution (Ubuntu 20.04 LTS or newer recommended)
- Administrative (sudo) privileges required

Software Requirements:

- **GNS3:** Graphical Network Simulator-3, a network simulation platform for simulating realistic network topologies
- **Docker:** Container system for VPN endpoints
- **Wireshark:** Network protocol analyzer

Hardware Requirements:

- At least 4GB RAM (8GB highly recommended for high performance)
- 20GB free disk space
- Network interface with Internet connectivity

Software Installation:
The following commands can be used to install the necessary software on Ubuntu/Debian systems:

```
# Install GNS3 and dependencies
sudo apt install gns3-gui gns3-server

# Install Docker
sudo apt install docker.io

# Install network analysis tools
sudo apt install wireshark

# Add user to required groups
sudo usermod -aG docker $USER
sudo usermod -aG wireshark $USER
```

Note: An install and login cycle may be required after installation in order for group membership changes to take effect. The VPN client programs (strongSwan for IPSec and OpenVPN for TLS) will be pre-installed inside the Docker containers during lab setup.

1.4 Lab Structure

The report is organized into these sections:

1. **Theoretical Background:** Introduction to VPN technologies, SoftEther VPN architecture, and employed security protocols.
2. **Network Topology:** Schematic network description employed during simulation and IP addressing strategy.
3. **Initial Configuration:** Setup of the GNS3 environment, the routers, and the container infrastructure.
4. **SoftEther VPN Server:** Installation of the VPN server with multi-protocol support.
5. **IPSec Configuration:** StrongSwan-based IPSec-based VPN implementation.
6. **TLS Configuration:** OpenVPN client-based TLS/SSL VPN implementation.

Each section contains in-practice implementation steps, configuration examples, and result analysis, providing a thorough overview of VPN technologies and applications in real-world contexts.

2 Theoretical Background

This chapter gives the theoretical background necessary to understand the Virtual Private Network technologies employed in this lab. We cover the fundamentals of VPNs, the specific architectures of SoftEther VPN, and the security protocols that enable the secure communication over untrusted networks.

2.1 Virtual Private Networks (VPNs)

Virtual Private Networks are network security solutions that provide connectivity utilizing shared infrastructure while maintaining properties typically offered in private networks. VPNs create secure communication channels over potentially untrusted networks by implementing various security mechanisms at different layers of the network stack.

Core VPN Properties:
A VPN must provide some essential properties to enable secure communication:

- **Data Confidentiality:** Encryption algorithms protect data content from unauthorized access during transmission over public networks.
- **Data Integrity:** Cryptographic hash functions and message authentication codes ensure that data has not been modified during transmission.
- **Authentication:** Both peer authentication (verification of communicating endpoints's identities) and data origin authentication are essential for secure VPN operation.
- **Access Control:** VPN policies determine which traffic is permitted and how it needs to be processed.
- **Anti-Replay Protection:** Sequence numbers and time stamps prevent malicious replay of previously captured packets.

VPN Classification:
VPNs can be classified based on their method of implementation and the network layers at which security is enforced. The layers considered in this lab are:

- **Network Layer VPNs:** Operate at the IP layer (L3), e.g. IPSec-based ones
- **Transport Layer VPNs:** Implement security at L4, e.g. TLS/SSL VPNs

2.2 SoftEther VPN Architecture

SoftEther VPN represents a unique approach to VPN deployment with a unified platform to support multiple VPN protocols simultaneously [1]. This feature of multi-protocol support enables straightforward comparison and evaluation of different VPN technologies in a single deployment.

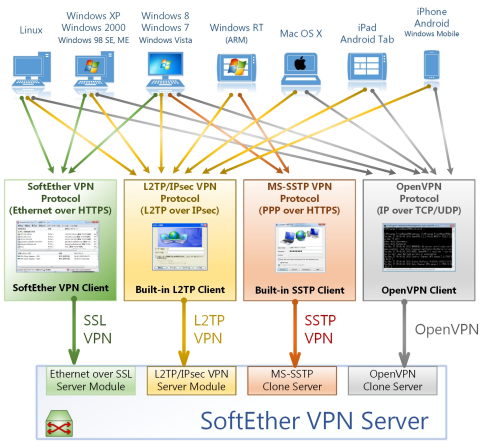

Figure 1: SoftEther VPN Multi-Protocol Architecture

As indicated in Figure 1, SoftEther VPN's architecture reflects its adaptability in the sense of offering Multiple client operating systems and VPN protocols. The figure explains how different hosts with different operating systems can connect to a single SoftEther VPN Server using different VPN protocols:

- **SoftEther VPN Protocol:** Ethernet over HTTPS Proprietary protocol
- **L2TP/IPSec VPN Protocol:** Common L2TP over IPSec implementation
- **MS-SSTP VPN Protocol:** Microsoft's Secure Socket Tunneling Protocol using PPP over HTTPS
- **OpenVPN Protocol:** SSL/TLS-based VPN using IP over TCP/UDP

Multi-Protocol Support:
SoftEther VPN's architecture relies on *protocol abstraction*, where different VPN protocols are realized as modules within a common framework. The key protocols that we will use in this lab are:

- **IPSec Support:** Built in support for both AH (Authentication Header) and ESP (Encapsulating Security Payload) protocols in tunnel mode
- **SSL/TLS Support:** Built-in TLS server compatible with OpenVPN clients and other SSL VPN solutions

Virtual Hub Architecture:
The core of SoftEther VPN is the Virtual Hub concept, which acts as a virtual Ethernet switch. This architecture facilitates the aggregation of multiple VPN sessions of different protocols, provides strong routing and bridging capabilities, and accomodates centralized policy enforcement and logging. Building complex network topologies, including both site-to-site networks and remote access networks, is also supported by the Virtual Hub and thus is a valuable component of SoftEther VPN applications.

SecureNAT Functionality:
SoftEther VPN comes with a Built-in NAT (Network Address Translation) and DHCP server feature named SecureNAT, which is employed to make it easy to set up VPN clients by

- Automatically assigning IP addresses to VPN clients
- Providing DNS resolution services
- Providing Internet access for VPN clients through NAT

2.3 IPSec Protocol Suite

Internet Protocol Security (**IPSec**) is a suite that provides security at the IP layer by adding headers to IP packets [2]. It may operate in transport mode (only securing payload) or tunnel mode (encapsulating the entire packet). IPSec implementations have been the subject of extensive cryptographic analysis [3] and are standardized by various bodies, like NIST [4].
A key part is the **Encapsulating Security Payload** (ESP), which ensures confidentiality by encrypting packets and provides optionally integrity and authentication as well. ESP in tunnel mode, encrypts the whole IP packet, creating new IP headers for the routing. **Security Associations** (SAs) are agreements between peers that specify parameters as encryption algorithms, keys, and endpoints; each of them is associated with a Security Parameter Index (SPI). Their manual management is usually controlled by Internet Key Exchange protocols: IKEv1 (with main and aggressive modes) and IKEv2 (which both improves performance and security). In this project, IPSec (via ESP, SA, and IKE) is used to establish an encrypted network layer tunnel for situations where protection at the kernel level is required for packets.

2.4 TLS/SSL for VPN Implementation

Transport Layer Security (TLS) provides session-layer security though a handshake to agree on cipher suites [5]. TLS-based VPNs (such as **OpenVPN**), tunnel arbitrary IP traffic in a TLS channel [6], leveraging the widespread availability of TLS support and negotiable cipher suites. This approach places security processing in user space, which may be easier to deploy and customize, but with some performance trade-offs compared to kernel-based solutions.

Layers of Authentication of TLS VPNs: Care must be taken to distinguish between TLS handshake authentication (typically server certificate verification) and application-layer authentication occurring inside the established TLS tunnel. User credentials are transmitted as application data after the secure channel has been established, again controlling access beyond the transport-layer security.

2.5 Client-Side VPN Technologies

StrongSwan (IPSec):
strongSwan is an open-source IPSec implementation used in this project for establishing and maintaining IPSec tunnels [7,8]. It supports IKEv1 and IKEv2, a cryptographic algorithm set, and other features like NAT traversal [9]. It can be configured with simple files (e.g., `ipsec.conf` and `ipsec.secrets`), and it supports the Linux kernel's IPSec stack. In our setup, strongSwan handles automatic key exchanges and maintains SAs for secure network-layer tunnels.

OpenVPN Client (TLS):
OpenVPN is the TLS-based client solution in this project [6,10]. It is based on OpenSSL for cryptography [11], supports both UDP and TCP transports, and can be configured in tun (Layer 3) or tap (Layer 2) mode. Certificate-based authentication using a certificate ensures that only authorized endpoints connect. OpenVPN's user-space implementation supports flexible scripting hooks and logging, so it is easily adapted to custom remote-access applications in the lab.

2.6 Comparative Analysis Framework

Having theoretical knowledge of the differences between IPSec and TLS implementations, provides the basis for effective comparison:

- **Architecture Differences:** IPSec is implemented at the network layer (typically in kernel space), whereas TLS-based VPNs run in user space at the session layer.
- **Performance Considerations:** IPSec generally has lower overhead with kernel integration; TLS-based VPNs offer more flexibility but may have more user space processing.
- **Deployment Complexity:** IPSec configurations may be more complex but follow broadly enstablished standards, while TLS-based VPNs like OpenVPN are more likely to have simpler initial setup and greater customizability.

This conceptual foundation establishes the necessary knowledge for understanding the actual application and comparison of SoftEther VPN's multi-protocol capabilities in subsequent parts of this lab activity.

3 Network Topology and Architecture

This section introduces the network topology employed for the SoftEther VPN lab activity. The setup is an actual simulation where two geographically separated private networks exchange data securely across the Internet via VPN tunnels.

3.1 GNS3 Network Simulation Platform

The lab setup is implemented using **GNS3 (Graphical Network Simulator-3)**, a full-fledged network simulation environment that can simulate intricate virtual network topologies [12]. GNS3 provides an all-in-one network learning, testing, and experimentation environment through the combination of multiple virtualization technologies into one GUI.

Key GNS3 Features:

GNS3 combines a range of important technologies to provide realistic network simulation:

- **Router Emulation:** Supports many Routers images emulation, providing authentic router behavior and command-line interfaces
- **Container Integration:** Native support for Docker container enables the deployment of Linux-based network applications and services.
- **Virtual Machine Support:** Hypervisor support for VirtualBox and VMware to run complete operating systems
- **Switch Simulation:** Integrated Ethernet switching to create sophisticated network topologies
- **Traffic Analysis:** Integrated packet capture with Wireshark for real-time network analysis

GNS3 provides simulation of geographically sparse networks connected over Internet infrastructure, ideal in this VPN lab to learn about VPN technology without requiring several locations or sophisticated hardware setups [13].

3.2 Topology Overview

The lab topology of the network consists of five major components that come together to generate a real-world Internet-based VPN scenario:

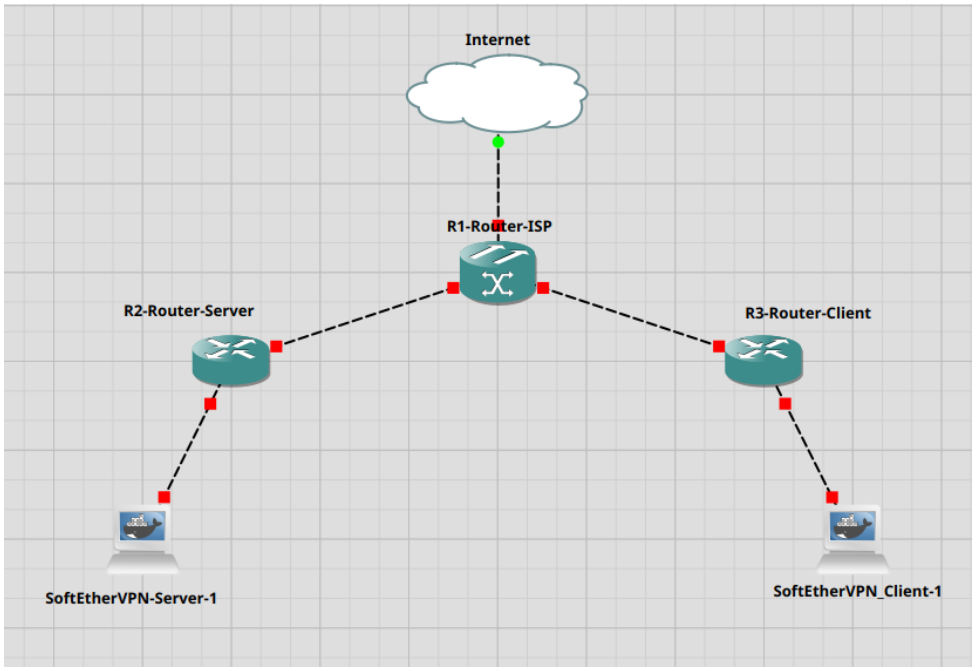

Figure 2: GNS3 Network Topology - VPN Across the Internet

The network topology employs an end-to-end VPN approach where:

- **Server Site:** Contains the SoftEther VPN server operating within a private network
- **Client Site:** Represents a remote branch office that requires VPN connectivity
- **Internet Infrastructure:** Virtual ISP network providing public connectivity
- **Edge Routers:** Perform NAT and routing functions, providing Internet connectivity for the private networks
- **VPN Endpoints:** Docker containers hosting the actual VPN software

This topology adequately demonstrates how end-to-end VPN technology enables secure communication between hosts of different private networks through untrusted public infrastructure, simulating common real-world deployment conditions where the VPN software runs directly on the endpoints rather than on gateway devices.

3.3 IP Addressing Scheme

The network address scheme uses a combination of public and private IP addresses to simulate realistic Internet connectivity. Addressing is according to RFC standards for private addressing (RFC 1918) and utilizes documentation addresses (RFC 5737) to simulate public IP.

Table 1: Network Interface Configuration

Device	Interface	IP Address	Description
Router 2 (Server-side)	LAN (Fa0/0)	10.0.1.1/24	Private Network 1
	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
	Interface Fa0/1	198.51.100.254/24	Connected to Router 3
	Interface Fa1/0	DHCP Assigned	Internet Cloud
Router 3 (Client-side)	WAN (Fa0/1)	198.51.100.1/24	Public IP (RFC 5737)
	LAN (Fa0/0)	10.0.2.1/24	Private Network 2
SoftEther Server	eth0	10.0.1.2/24	VPN Server Host
VPN Client	eth0	10.0.2.2/24	VPN Client Host

Network Segments:

- **Server Network (10.0.1.0/24):** Private subnet for hosting the SoftEther VPN server
- **Client Network (10.0.2.0/24):** Remote private subnet for VPN client
- **ISP Segment 1 (203.0.113.0/24):** Public network between ISP and server-side
- **ISP Segment 2 (198.51.100.0/24):** Public network between ISP and client-side
- **VPN Tunnel Networks:** Virtual addressing used for VPN connectivity (dynamically allocated)

3.4 Device Roles and Functions

Every device of the network topology is responsible for a particular role in demonstrating VPN functionality:

3.4.1 ISP Router (R1-Router-ISP)

The ISP router simulates Internet Service Provider infrastructure and provides:

- **Internet Connectivity:** Public IP addressing allocated though DHCP
- **Inter-Site Routing:** Routes traffic among the remote sites by utilizing simulated infrastructure
- **NAT Traversal Support:** Enables VPN protocols to work over NAT.

- **Public IP Assignment:** Provides public addressing for both edge routers

The ISP router configuration contains static routes for both private networks and uses a default route towards the Internet cloud.

3.4.2 Server-Side Router (R2-Router-Server)

This edge router connects the private network of the server to the Internet and hosts:

- **Network Address Translation:** Port Address Translation management
- **VPN Port Forwarding:** Static NAT rules enable forwarding of VPN traffic to the server:
  - Port 500/UDP: ISAKMP forwarded to SoftEther server
  - Port 4500/UDP: NAT-T forwarded to SoftEther server
  - Port 443/TCP: HTTPS/TLS forwarded to SoftEther server
- **Default Gateway:** Routing for the server's private network
- **Transparent Routing:** Routes VPN traffic between Internet and private network without VPN processing

3.4.3 Client-Side Router (R3-Router-Client)

The client-side edge router fulfills the same role for the remote site:

- **PAT Configuration:** Network address translation for client network connectivity
- **Default Routing:** Internet connectivity for the client private network
- **Transparent VPN Support:** Allows VPN client software to connect using NAT without router-based VPN processing

3.4.4 SoftEther VPN Server Container

The server container runs the SoftEther VPN software with the following parameters:

- **Container Image:** `siomiz/softethervpn:latest`
- **Multi-Protocol Support:** Concurrent IPSec and TLS/SSL VPN services
- **Virtual Hub:** DEFAULT hub for client connections
- **User Management:** Set with test user credentials for VPN authentication
- **Certificate Management:** Self-signed certificates for TLS-based connections

3.4.5 VPN Client Container

The client container simulates a remote user or branch office with:

- **Container Image:** `ubuntu:latest`
- **IPSec Client:** StrongSwan software for IPSec connectivity
- **TLS Client:** OpenVPN client for SSL/TLS VPN connections

3.5 Network Communication Flow

The network design supports diverse communication scenarios:

1. **Direct Internet Communication:** Both sites can access Internet resources through their respective ISP connections
2. **End-to-End VPN Tunnel Establishment:** The VPN client container establishes direct VPN connections to the SoftEther server container using either:
   - IPSec protocol (strongSwan client to SoftEther server)
   - TLS/SSL protocol (OpenVPN client to SoftEther server)
3. **Encrypted Inter-Site Communication:** Once the VPN tunnels are created, the client will be able to communicate securely with resources in the server's network
4. **NAT Traversal:** VPN protocols provide automatic NAT traversal, enabling encrypted tunnels to traverse through the edge routers

4 Initial Network Configuration

This section describes the step-by-step laboratory environment setup process, including GNS3 setup, router configuration, and Docker container deployment. The configuration creates the underlying network infrastructure to be utilized by the VPN implementations.

4.1 GNS3 Environment Setup

Here we detail the technical steps required for implementing the network topology presented in Section 3. All device connections and configuration follow architecture specs provided in the GNS3 Network Simulation Platform subsection.

4.1.1 Device Import and Template Creation

Cisco Router Template:
We have chosen for this activity the Cisco 7200 router running IOS version 124-24.T5. This specific IOS image is chosen because it is a freely downloadable equivalent of modern commercial Cisco IOS releases. Whilst newer platforms like IOSv (IOS Virtual) are faster and more feature-rich, they require valid Cisco license agreements that are too costly for learning purposes. The 124-24.T5 image, although older and not usually found in modern production settings, has all the fundamental routing and NAT functionality for this lab exercise. To import the router template:

1. Open GNS3 and click the **router icon** in the left panel
2. Select **"New Template"** at the bottom
3. Choose **"Install an appliance from the GNS3 server"**
4. Filter by **"CISCO"** and locate **"Cisco 7200"**
5. Install the template and select version **"124-24.T5"**
6. Load the router image file to complete the setup

Docker Container Setup:
Two Docker containers are required for the VPN endpoints [14]:

- **Server Container Image:** `siomiz/softethervpn:latest` [15]
- **Client Container Image:** `ubuntu:latest` [16]

To insert containers in GNS3 project:

1. Navigate to **Edit → Preferences** in GNS3
2. Select **"Docker containers"** from the left panel
3. Click **"New"** to add a container
4. Enter the image name (e.g., `siomiz/softethervpn:latest`)
5. Configure the container name and complete the setup
6. Do the same for the Ubuntu client container

You should be able to see the containers in the left panel of GNS3 now, ready to be pulled into the workspace. So now you can configure the same topology as shown in Figure 2 in Section 3 of this document.
Clicking on the wire on the left panel, you can connect the devices to each other, creating the network topology described. Choose the correct interfaces for each link as shown in the table in Section 3.2. The interfaces connected from the Client router and Server one to the internal nodes, are always the FastEthernet0/0, instead the FastEthernet0/1 one is the one connected to the ISP. The ISP with FastEthernet0/0 is connected to the server router, 0/1 is connected instead to the client router. There is, here besides, another interface, the one of FastEthernet1/0, which will be instead plugged into the

internet bridge, so with the cloud and directly to the right interface of the host machine that provides the internet connection.
Critical Router Configuration Note: Before proceeding with the router configurations, you must configure the ISP router to have three interfaces instead of the default two. This is needed because the ISP router must connect to both edge routers and the Internet cloud. To perform the ISP router configuration:

1. Right-click on the ISP router in GNS3
2. Select **"Configure"** from the context menu
3. Navigate to the **"Slots"** section in the configuration dialog
4. Replace the existing slot configuration from **"C7200-IO-FE"** to **"C7200-IO-2FE"**
5. Apply the changes and close the configuration dialog

This configuration change provides the ISP router with three FastEthernet interfaces (Fa0/0, Fa0/1, and Fa1/0) as required by the network topology, allowing it to connect to both edge routers and the Internet cloud simultaneously.

4.2 Container Configuration

The Docker containers must have special configuration for persistent network and storage setup.

4.2.1 Server Container Configuration

SoftEther VPN server container needs permanent directories to store configuration files:

1. Open the server container configuration in GNS3 (right-click on the container)
2. Navigate to **Advanced Settings**
3. Add the following additional directory:
    - `/usr/vpnserver`

This directory will be created in the GNS3 project directory on the local system, which will allow persistent storage of configuration files during restart of the container.
Server Configuration File Setup:
In order to ensure the SoftEther VPN server starts with the proper configuration, you should place the `vpn_server.config` file in the correct persistent directory:

1. Navigate to your local file system GNS3 project directory
2. Open the `project-files` folder
3. Look for the `docker` subdirectory
4. Find the server container folder by means of the container ID
5. Enter into the `usr/vpnserver` directory within the container folder
6. Place the `vpn_server.config` file there

This will enable the pre-configured settings, like user accounts, IPSec setup, and SSL/TLS configuration, to be loaded automatically when the container starts as started by the SoftEther VPN service.

4.2.2 Client Container Setup

The client container requires similar setup for persistent storage:

1. Open the client container configuration in GNS3
2. Navigate to Advanced Settings
3. Add the additional directory: `/client`

Client Configuration Files Setup:
The client container requires a number of configuration files for IPSec and TLS VPN implementations. Place all the following files in the client's persistent directory:

1. Navigate to your GNS3 project directory on the local filesystem
2. Open the `project-files` folder
3. Locate the `docker` subdirectory
4. Find the client container folder
5. Navigate to the `client` directory within the container folder
6. Place the following files in this directory:
   - **IPSec files:** `ipsec.conf` and `ipsec.secrets`
   - **TLS/OpenVPN files:** `softether.ovpn`, `ca.crt`, and `credentials.txt`

This setup ensures that all VPN client configuration setups are available when the container boots in order to create smooth connections with both IPSec and TLS VPN servers.

4.3 Project Startup and Device Access

After each of the configurations is completed and files are properly placed inside their persistent folders, you can initiate the lab environment.

4.3.1 Launching the GNS3 Project

To launch all devices and containers in the project:

1. Click the green **Play button** in the top toolbar of GNS3
2. This will boot every Docker container and router simultaneously
3. Wait until all the devices complete their boot sequence
4. Ensure that all icons of the devices have a green status icon

4.3.2 Accessing Device Terminals

In order to talk to network devices and containers:

For Cisco Routers:

1. Right-click on any router device
2. Select **"Console"** from the context menu
3. This gives access to the router's command-line interface

For Docker Containers:

1. Right-click on any container (server or client)
2. Choose **"Auxiliary Console"** from the context menu
3. This opens the container's terminal interface
4. Use this terminal for all commands and configurations related to containers

Key Note: For Docker containers, always use the "Auxiliary Console" option and not the standard console, as it gives the correct terminal interface to work with the containerized operating system.

4.4 Router Configuration

Now that you have access to the router terminals, go ahead and configure the three routers with individual configurations to mimic realistic Internet connectivity and routing behavior.

4.4.1 ISP Router (R1-Router-ISP)

The ISP router provides Internet connectivity and inter-site routing. The setup includes interface setup, MAC address spoofing for DHCP, and routing table entries.

```
enable
configure terminal

# Configure interface to Router 2 (Server-side)
interface FastEthernet0/0
  ip address 203.0.113.254 255.255.255.0
  no shutdown
exit

# Configure interface to Router 3 (Client-side)
interface FastEthernet0/1
  ip address 198.51.100.254 255.255.255.0
  no shutdown
exit

# Configure Internet interface with DHCP and MAC spoofing
interface FastEthernet1/0
  mac-address xxxx.xxxx.xxxx  # Replace with host machine MAC
  ip address dhcp
  no shutdown
exit

# Configure static routes for private networks
ip route 198.51.100.0 255.255.255.0 FastEthernet0/1
ip route 203.0.113.0 255.255.255.0 FastEthernet0/0

# Configure default route to Internet
ip route 0.0.0.0 0.0.0.0 FastEthernet1/0

end
write memory
```

Important Note - MAC Address Spoofing:
This step is required only needed when your host's DHCP server restricts MAC addresses (e.g., Politecnico di Torino WiFi). In other cases, omit this process or adapt accordingly.
The MAC address spoofing is required since the Politecnico di Torino WiFi network's DHCP server uses MAC address filtering to ensure security. Without the use of the host machine's MAC address, the DHCP server will not respond with a valid IP address for the router interface, which denies Internet access for the virtualized network. To know your host machine's MAC address:

```
# Display all network interfaces and their MAC addresses
ip a

# Look for the active network interface (usually wlan0 for WiFi or eth0 for
    Ethernet)
# The MAC address appears after "link/ether"
# Example output:
# 2: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
    default qlen 1000
#     link/ether 1c:ce:51:3f:98:20 brd ff:ff:ff:ff:ff:ff
```

Copy the MAC address from your active network interface and convert it to Cisco format by adding dots every four characters:

- **Linux format:** aa:bb:cc:dd:ee:ff

- **Cisco format:** aabb.ccdd.eeff

Replace `xxxx.xxxx.xxxx` in the configuration with your converted MAC address.

4.4.2 Server-Side Router (R2-Router-Server)

This router connects the server's private network to the Internet and implements NAT with port forwarding for VPN services.

```
enable
configure terminal

# Configure LAN interface (connected to server)
interface FastEthernet0/0
  ip address 10.0.1.1 255.255.255.0
  ip nat inside
  no shutdown
exit

# Configure WAN interface (connected to ISP)
interface FastEthernet0/1
  ip address 203.0.113.1 255.255.255.0
  ip nat outside
  no shutdown
exit

# Configure static NAT for VPN services
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

# Configure PAT for general Internet access
access-list 1 permit 10.0.1.0 0.0.0.255
ip nat inside source list 1 interface FastEthernet0/1 overload

# Configure default route
ip route 0.0.0.0 0.0.0.0 FastEthernet0/1

end
write memory
```

Port Forwarding Explanation:

- **Port 500/UDP:** ISAKMP (Internet Security Association and Key Management Protocol)
- **Port 4500/UDP:** NAT-T (NAT Traversal for IPSec)
- **Port 443/TCP:** HTTPS/TLS for SSL VPN connectivity

4.4.3 Client-Side Router (R3-Router-Client)

The client-side router provides NAT and routing for the client's private network.

```
enable
configure terminal

# Configure LAN interface (connected to client)
interface FastEthernet0/0
  ip address 10.0.2.1 255.255.255.0
  ip nat inside
  no shutdown
exit

# Configure WAN interface (connected to ISP)
interface FastEthernet0/1
```

```
  ip address 198.51.100.1 255.255.255.0
  ip nat outside
  no shutdown
exit

# Configure default route
ip route 0.0.0.0 0.0.0.0 FastEthernet0/1

# Configure PAT for Internet access
access-list 1 permit 10.0.2.0 0.0.0.255
ip nat inside source list 1 interface FastEthernet0/1 overload

end
write memory
```

4.5 Container Network Configuration

Now that you have access to the container terminals, configure the network interfaces for both containers.

4.5.1 Server Network Configuration

Configure the server container's network interface:

```
# Configure IP address and default route
ip addr add 10.0.1.2/24 dev eth0
ip route add default via 10.0.1.1

# Verify network configuration
ip addr show
ip route show

# Test connectivity to gateway
ping 10.0.1.1
```

4.5.2 Client Network Configuration

Configure the client container's network interface and install required VPN software:

```
# Configure network interface
ip addr add 10.0.2.2/24 dev eth0
ip route add default via 10.0.2.1

# Update package repositories
apt update

# Install strongSwan for IPSec VPN
apt install strongswan -y

# Install OpenVPN for TLS VPN
apt install openvpn -y

# Copy configuration files from persistent storage
cp /client/ipsec.conf /etc/
cp /client/ipsec.secrets /etc/

# Verify network configuration
ip addr show
ip route show

# Test connectivity to gateway
ping 10.0.2.1
```

4.6 Basic Connectivity Testing

Before proceeding with VPN configuration, verify that the basic network infrastructure is functioning correctly.

4.6.1 Inter-Router Connectivity

Test connectivity between routers to ensure proper routing:

```
# From ISP Router - test connectivity to edge routers
ping 203.0.113.1    # Should reach Server-side router
ping 198.51.100.1   # Should reach Client-side router

# From Server-side Router - test connectivity to ISP
ping 203.0.113.254  # Should reach ISP router

# From Client-side Router - test connectivity to ISP
ping 198.51.100.254 # Should reach ISP router
```

4.6.2 End-to-End Connectivity

Test connectivity between the container endpoints:

```
# From Server Container - test connectivity to public IPs
ping 198.51.100.1   # Should reach Client-side router public IP

# From Client Container - test connectivity to public IPs
ping 203.0.113.1    # Should reach Server-side router public IP
```

4.6.3 Service Verification

Verify that the SoftEther VPN server is running and listening on required ports:

```
# Check if SoftEther VPN server is listening
ss -tuln | grep -E '(443|500|4500)'
```

4.7 Network Infrastructure Verification

Now, the fundamental network infrastructure must be up and running with:

- Three routers configured with appropriate IP addressing and routing
- NAT services functioning on edge routers
- Docker containers with network connectivity
- SoftEther VPN server operational and accessible
- Straightforward inter-site connectivity through emulated Internet

This configuration offers the foundation for deployment of VPN services described in the subsequent sections. Any connectivity issues at this point must be resolved before configuring VPN because they will prevent successful creation of VPN tunnels.
The next section will describes how to configure the SoftEther VPN server to provide multi-protocol VPN services for both IPSec and TLS-based connections.

5 SoftEther VPN Server Configuration

This section explains the deployment and installation of the SoftEther VPN server within the Docker container environment.

5.1 Server Setup and Installation

The SoftEther VPN server runs in a Docker container from the pre-built image `siomiz/softethervpn:latest`. This containerized approach has various advantages like isolation, portability, and simplicity of deployment.

5.1.1 Container Deployment

The server container has been configured in GNS3 with persistent storage folders as explained in Section 4. Once the container is running, the SoftEther VPN service automatically starts with the following features:

- **Automatic Service Start:** The VPN server daemon automatically runs when the container starts, due to the persistent storage setup, the server will be already configured with the correct configuration file.
- **Multi-Protocol Listeners:** Support for IPSec, SSL/TLS simultaneously
- **Network Integration:** Automatic integration with the container's network interface (eth0)

5.1.2 Service Verification

To verify that the SoftEther VPN server operates correctly, execute the following commands from within the server container:

```
# Check if SoftEther VPN server process is running
ps aux | grep vpnserver

# Verify VPN server is listening on required ports
ss -tuln | grep -E '(443|500|4500|5555)'

# Expected output should show listeners on:
# tcp LISTEN 0.0.0.0:443   (HTTPS/SSL VPN)
# udp LISTEN 0.0.0.0:500   (ISAKMP/IKE)
# udp LISTEN 0.0.0.0:4500  (NAT-T)
# tcp LISTEN 0.0.0.0:5555  (Management/API)
```

5.2 Configuration File Analysis

The SoftEther VPN server configuration is configured by the `vpn_server.config` file, with detailed settings for all VPN protocols, virtual hubs, user administration, and security policies.

5.2.1 Core Server Settings

Server setup consists of several core elements

```
# Core server configuration parameters
declare ServerConfiguration
{
    # Accept non-TLS connections
    bool AcceptOnlyTls false
    # Default cipher suite
    string CipherName DHE-RSA-AES256-SHA
    # Allow IPSec aggressive mode
    bool DisableIPsecAggressiveMode false
    # Enable NAT traversal
    bool DisableNatTraversal false
```

```
    # Enable OpenVPN compatibility
    bool DisableOpenVPNServer false
    # Connection limit per IP
    uint MaxConnectionsPerIP 256
    # Enable debug logging
    bool SaveDebugLog true
}
```

Key Configuration Parameters:

- **Multi-Protocol Support:** All significant VPN protocols are supported by default
- **NAT Traversal:** Should be enabled for clients behind NAT devices
- **Security Settings:** Secure cipher suites with DHE for perfect forward secrecy
- **Connection Limits:** Reasonable connection limits to prevent resource exhaustion
- **Logging:** Comprehensive logging for troubleshooting and analysis

5.2.2 Protocol Listener Configuration

The server establishes multiple listeners for a range of VPN protocols:

```
declare ListenerList
{
    declare Listener0  # HTTPS/SSL VPN
    {
        bool Enabled true
        uint Port 443
    }
    declare Listener1  # ISAKMP/IKE
    {
        bool Enabled true
        uint Port 500
    }
    declare Listener2  # NAT-T
    {
        bool Enabled true
        uint Port 4500
    }
    declare Listener3  # Management/API
    {
        bool Enabled true
        uint Port 5555
    }
}
```

Protocol Listener Details:

- **Port 443/TCP:** HTTPS/SSL VPN for OpenVPN-compatible connections
- **Port 500/UDP:** ISAKMP/IKE for initial IPSec negotiation
- **Port 4500/UDP:** NAT-T (NAT Traversal) - Essential for IPSec operation behind NAT device.
- **Port 5555/TCP:** Management and API interface for server administration

5.3 Virtual Hub and User Management

SoftEther VPN handles connections with Virtual Hubs, which act as virtual Ethernet switches. The default setting includes a single hub named "DEFAULT" with basic user authentication.

5.3.1 Default Virtual Hub Configuration

The DEFAULT hub provides the following services:

- **User Authentication:** basic Password-based authentication for VPN clients (not using the cerificate authentication for the mutual authentication)
- **SecureNAT:** In-built DHCP and NAT services for allocating IP to clients
- **Access Control:** Traffic filtering and routing policies can be configured

5.3.2 User Account Management

The server also includes a test user account for VPN authentication:

```
declare UserList
{
    declare user1
    {
        # Password: "ciao" (hashed)
        byte AuthPassword ObNWU1DckHL0Xg4HuyRAMKiIANY=
        # Password authentication
        uint AuthType 1
        # No additional notes
        string Note $
    }
}
```

Important Authentication Note: The user credentials (user1/ciao) are not used during the TLS handshake process itself. Instead, the credentials are used for an application-layer authentication that is performed after successful establishment of the TLS tunnel. The TLS handshake only authenticates the server certificate, while the user authentication happens at a higher layer within the securely established tunnel.

5.4 IPSec Protocol Configuration

The SoftEther VPN server already has IPSec built-in, so a normal IPSec client can be connected without extra software. Pre-shared key is not a strong one here, but it works for lab environments.

5.4.1 IPSec Settings

```
declare IPsec
{
    # Pre-shared key for IPSec
    string IPsec_Secret ciao
    # Default hub for L2TP connections
    string L2TP_DefaultHub DEFAULT
    # Disable L2TP/IPSec (using native IPSec)
    bool L2TP_IPsec false
    # Disable raw L2TP
    bool L2TP_Raw false
}
```

IPSec Configuration Details:

- **Pre-Shared Key:** ”ciao” - used for IPSec authentication, usage of a weak key only for lab purposes
- **Default Hub:** All IPSec connections are directed to the DEFAULT virtual hub
- **Protocol Mode:** Native IPSec implementation rather than L2TP/IPSec combination

5.5 SSL/TLS Protocol Configuration

The server provides SSL/TLS VPN services compatible with OpenVPN clients and other SSL VPN solutions.

5.5.1 TLS Settings and Certificates

The server uses self-signed certificates for TLS connections:

```
# TLS configuration parameters
string OpenVPNDefaultClientOption dev-type$20tun,link-mtu$201500,tun-mtu$201500,
    cipher$20AES-128-CBC,auth$20SHA1,keysize$20128,key-method$202,tls-client
```

SSL/TLS Features:

- **OpenVPN Compatibility:** Full compatibility with standard OpenVPN clients
- **Cipher Support:** AES-128-CBC encryption with SHA1 authentication. Usage of a weak hash function only for lab purposes, to increase the security, it is recommended to use SHA256 or stronger, modifying accordingly the other parameters.
- **TUN Interface:** Layer-3 tunneling for IP packet forwarding
- **Certificate-Based Authentication:** X.509 certificate validation for enhanced security

5.6 SecureNAT Configuration

SecureNAT provides integrated DHCP and NAT services for VPN clients, simplifying client configuration and enabling Internet access.

SecureNAT Benefits:

- **Automatic IP Assignment:** Clients receive IP addresses from 192.168.30.10-200 range
- **DNS Services:** Integrated DNS resolution for VPN clients
- **Internet Access:** NAT functionality enables clients to access external resources
- **Simplified Configuration:** Clients require minimal manual network configuration

5.6.1 Server Certificates

The server uses self-signed X.509 certificates for TLS operations:

- **Certificate Subject:** da3af5075c51 (unique identifier)
- **Key Length:** 2048-bit RSA
- **Validity Period:** Long-term validity for lab use
- **Usage:** Server authentication for TLS/SSL connections

5.6.2 Security Policies

The server uses numerous security policies:

```
# Security-related settings
# Enable DoS protection
bool DisableDosProtection false
# Allow session reconnection
bool DisableSessionReconnect false
# DNS thread limit
uint MaxConcurrentDnsClientThreads 512
# Connection limit
uint MaxUnestablishedConnections 1000
# Send server signature
bool NoSendSignature false
```

Security Features:

- **DoS Protection:** Built-in protection against denial-of-service attacks
- **Connection Limits:** Prevents resource exhaustion through connection limiting
- **Session Management:** Secure session handling with reconnection support
- **Authentication:** Multiple authentication methods including certificates and passwords

5.7 Operational Verification

Ensure the server configuration supports both IPSec and SSL/TLS protocols:

- **IPSec Listeners:** Ports 500 (ISAKMP) and 4500 (NAT-T) are active
- **SSL/TLS Listeners:** Ports 443 is accepting connections
- **User Authentication:** Test user "user1" is configured and accessible
- **Virtual Hub:** DEFAULT hub is operational with SecureNAT enabled
- **NAT Forwarding:** Router forwarding rules are directing traffic correctly

The SoftEther VPN server is configured and ready to listen for IPSec and TLS/SSL-connections. The sections that follow describe the configuration of both of these client types and show secure tunnel establishment over the test Internet infrastructure.

6 IPSec VPN Configuration

This section details the configuration of the IPSec-based VPN connection using strongSwan as the client software. The IPSec implementation provides network-layer security with encryption and authentication, establishing a secure tunnel between the client container and the SoftEther VPN server.

6.1 strongSwan Client Setup

The strongSwan software was installed during the initial container configuration as part of Section 4. To verify the installation and ensure all components are available:

```
# Verify strongSwan installation
apt list --installed | grep strongswan

# Check available strongSwan tools
which ipsec
```

6.2 IPSec Configuration Files

strongSwan uses two primary configuration files: `ipsec.conf` for connection parameters and `ipsec.secrets` for authentication credentials.

6.2.1 ipsec.conf Analysis

The `ipsec.conf` file defines the IPSec connection parameters, including encryption algorithms, authentication methods, and tunnel endpoints.

```
# /etc/ipsec.conf
config setup
    charondebug="ike 2, knl 2, cfg 2"    # Debugging levels
    uniqueids=yes                        # Ensure unique IDs

conn softether
    # Local settings
    left=%any                           # Accept any local IP
    leftsubnet=10.0.2.0/24              # Client subnet to encrypt
    leftid=@client                      # Unique client identifier

    # Remote settings
    right=203.0.113.1                   # Server public IP
    rightid=10.0.1.2                    # Server's actual IP
    rightsubnet=0.0.0.0/0               # All traffic via VPN

    # NAT-Traversal
    forceencaps=yes                      # Force UDP encapsulation

    # Phase 1 (IKEv1)
    keyexchange=ikev1                    # Use IKEv1 protocol
    ike=aes256-sha1-modp2048!            # IKE encryption
    esp=aes128-sha1-modp1024!            # ESP encryption
    aggressive=no                        # Use main mode

    # Authentication
    leftauth=psk                         # Pwd authentication
    rightauth=psk                        # Server uses PSK too

    # Dead Peer Detection
    dpdaction=restart                    # Restart on peer failure
    dpddelay=30s                         # DPD check interval
    dpdtimeout=120s                      # DPD timeout

    auto=start                           # Auto-start connection
```

Configuration Parameter Explanation:

- **Connection Identity:** `left/right` parameters define local and remote endpoints
- **Subnet Configuration:** `leftsubnet` specifies which traffic should be encrypted
- **NAT Traversal:** `forceencaps=yes` ensures IPSec works through NAT devices
- **Algorithm Selection:** Encryption with AES-256 for IKE and AES-128 for ESP, also here the usage of a weak hash function (SHA1) is only for lab purposes.
- **Authentication:** Pre-shared key (PSK) method matching server configuration
- **Dead Peer Detection:** Automatic tunnel recovery on connection failure

6.2.2 ipsec.secrets Configuration

The `ipsec.secrets` file contains authentication credentials, specifically the pre-shared key that must match the server configuration.

```
# /etc/ipsec.secrets
# Format: local_id remote_id : PSK "shared_secret"

%any 203.0.113.1 : PSK "ciao"
```

Authentication Details:

- **Local ID:** `%any` accepts any local IP address
- **Remote ID:** `203.0.113.1` specifies the SoftEther server's public IP
- **Authentication Method:** PSK (Pre-Shared Key)
- **Shared Secret:** "ciao" - must match the server's IPSec_Secret configuration

Important Security Note: In production environments, pre-shared keys should be significantly more complex and randomly generated. The simple "ciao" key is used here for laboratory demonstration purposes only.

6.3 Connection Establishment

The IPSec tunnel establishment process involves multiple phases, including IKE negotiation and ESP Security Association creation.

6.3.1 Manual Connection Initiation

To establish the IPSec connection manually:

```
# Restart strongSwan to reload configuration
ipsec restart

# Initiate the VPN connection
ipsec up softether

# Verify connection status
ipsec status

# Check established Security Associations
ipsec statusall
```

6.3.2 Connection Verification

Successful IPSec tunnel establishment can be verified through multiple methods:

```
# Test connectivity through tunnel
ping 192.168.30.1  # SoftEther SecureNAT gateway

# Check routing table for VPN routes
ip route show
```

6.3.3 Debug and Troubleshooting

For troubleshooting connection issues, strongSwan provides comprehensive debugging:

```
# Start strongSwan with full debugging
ipsec start --nofork --debug-all

# Verify network connectivity to server
ping 203.0.113.1
```

6.4 Traffic Analysis

Understanding how IPSec encapsulates and processes network traffic is crucial for verification and troubleshooting.

6.4.1 Wireshark Packet Analysis

To observe IPSec traffic in detail, it is highly recommended to open a Wireshark instance on one of the network cables connecting the client and server infrastructure. This allows real-time analysis of the VPN establishment and data transmission phases.

IPSec Tunnel Establishment Phase:
During the initial IPSec connection setup, Wireshark will capture ISAKMP (Internet Security Association and Key Management Protocol) packets. You can start the Wireshark capture instance by right clicking on a wire in GNS3, and then selecting "Start Capture".

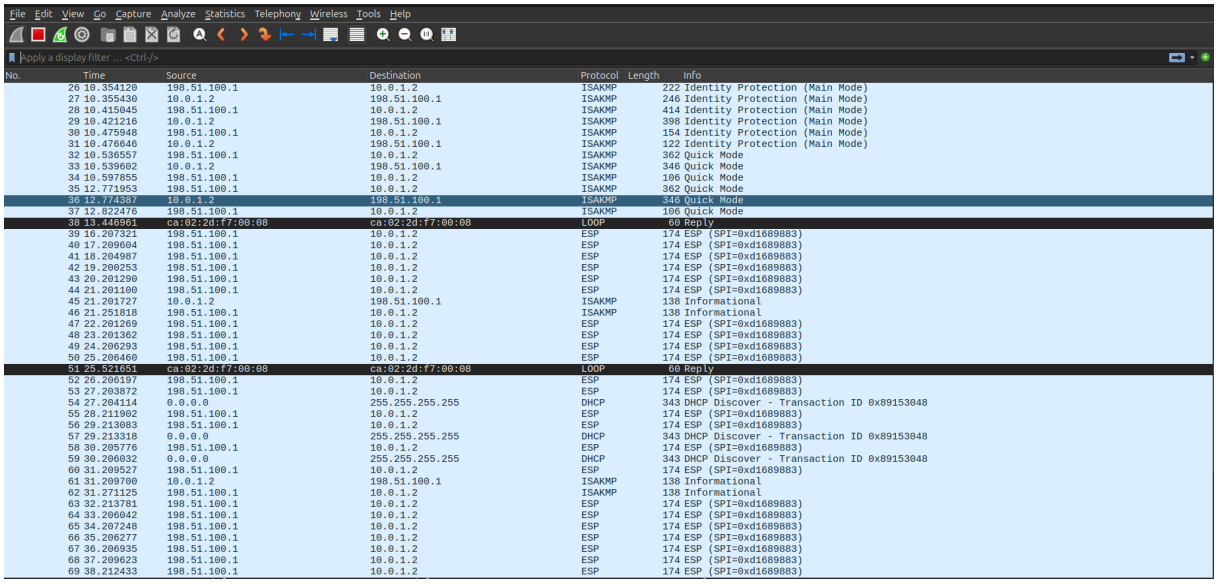

Figure 3: IPSec VPN Traffic Analysis - ISAKMP and ESP Protocol Exchange

Figure 3 demonstrates a comprehensive IPSec VPN session captured with Wireshark, showing the complete lifecycle of an IPSec tunnel establishment and data transmission. The capture reveals several distinct phases:

ISAKMP Phase 1 (Main Mode): The initial packets (frames 26-31) show the ISAKMP Main Mode negotiation on UDP port 500, where the VPN endpoints establish the first Security Association (SA) for protecting subsequent key exchange traffic. This phase includes:

- Identity Protection exchange for secure peer authentication
- Negotiation of encryption and authentication algorithms
- Diffie-Hellman key exchange for establishing shared secrets

ISAKMP Phase 2 (Quick Mode): Following the main mode, Quick Mode packets establish the Security Associations for actual data protection, negotiating the parameters for ESP (Encapsulating Security Payload) tunnels.

ESP Data Transmission: The latter portion of the capture shows ESP packets carrying encrypted user data. These packets demonstrate the IPSec tunnel in operation, where all original IP traffic is encapsulated within ESP headers and encrypted according to the negotiated algorithms.

Active Tunnel Communication Phase:
Once the IPSec tunnel is established and active communication begins, the packet capture will show ESP (Encapsulating Security Payload) packets:

```
# Filter for ESP traffic during active communication

# Test communication to generate ESP traffic
ping 192.168.30.1  # From client to SoftEther SecureNAT
```

During active communication, you will observe:

- **ESP packets:** Encrypted data transmission with visible ESP headers
- **Encrypted payload:** Data content protected by IPSec encryption
- **Tunnel endpoints:** Source and destination IPs showing the VPN gateway addresses

7 TLS/SSL VPN Configuration

This section details the configuration of the TLS-based VPN connection using OpenVPN as the client software. The TLS implementation provides session-layer security with certificate-based authentication and encryption, establishing a secure tunnel between the client container and the SoftEther VPN server.

7.1 OpenVPN Client Setup

OpenVPN is a robust and highly configurable VPN solution that uses SSL/TLS for key exchange and encryption. It operates in user space and provides flexibility in configuration and deployment.

7.1.1 Software Installation

The OpenVPN client software was installed during the initial container configuration. To verify the installation:

```
# Verify OpenVPN installation
openvpn --version

# Check OpenVPN capabilities
openvpn --show-ciphers | head -10
openvpn --show-digests | head -10
```

7.1.2 Certificate and Key Management

Unlike IPSec's pre-shared key approach, OpenVPN uses X.509 certificates for authentication, providing stronger security and better scalability.

Required Certificate Files:

- **CA Certificate:** Root certificate to verify server authenticity
- **Client Certificate:** Client's public key certificate (not used in this setup)
- **Server Verification:** Server certificate validation parameters
- **User Credentials:** Username and password for authentication

7.2 Certificate and Credential Management

The TLS-based VPN uses a combination of certificate-based server authentication and username/password client authentication.

7.2.1 User Credentials

The client authenticates using username and password stored in a credentials file:

```
# /client/credentials.txt
user1
ciao
```

Authentication Details:

- **Username:** user1 (matches SoftEther server user configuration)
- **Password:** ciao (matches server's user password)
- **Security:** File should have restricted permissions (600) in production
- **Authentication Layer:** These credentials are used for application-layer authentication after TLS handshake completion, not during the handshake itself

7.3 OpenVPN Configuration Analysis

The OpenVPN configuration file (*.ovpn) contains all parameters necessary for establishing the TLS VPN connection.

7.3.1 Complete Configuration File

```
# /client/softether.ovpn
client                                  # Client mode
dev tun                                 # TUN interface
proto tcp                               # TCP transport protocol
remote 203.0.113.1 443                  # Server IP and port
resolv-retry infinite                   # Retry DNS resolution
nobind                                  # Don't bind local port
persist-key                             # Keep keys in memory
persist-tun                             # Keep TUN interface
ca ca.crt                               # CA certificate file
remote-cert-tls server                  # Verify server cert
verify-x509-name da3af5075c51 name      # Verify server CN
auth-user-pass credentials.txt          # Username/password file
cipher AES-128-CBC                      # Encryption cipher
data-ciphers AES-128-CBC                # Allowed ciphers
mssfix 1450                             # MSS clamping for MTU
verb 4                                  # Verbosity level
```

Configuration Parameter Analysis:

- **Connection Type:** Client mode with TCP transport for reliability
- **Interface:** TUN device for Layer-3 IP tunneling
- **Server Details:** Connects to SoftEther server on port 443 (HTTPS)
- **Authentication:** Dual authentication with certificates and credentials
- **Encryption:** AES-128-CBC cipher for data encryption
- **Network Optimization:** MSS fixing to prevent fragmentation issues

7.3.2 Security Features

The OpenVPN configuration implements several security mechanisms:

1. **Server Authentication:** X.509 certificate ensures server identity during TLS handshake
2. **Certificate Name Verification:** CN prevents man-in-the-middle attacks during TLS handshake
3. **Strong Encryption:** AES-128-CBC provides confidentiality
4. **Transport Security:** TLS transport layer provides additional protection

Authentication Process Clarification: The TLS handshake process only involves server certificate validation. The user credentials (user1/ciao) are transmitted and validated within the established TLS tunnel as an additional application-layer authentication step, providing defense-in-depth security.

7.4 TLS Handshake and Connection

The TLS VPN establishment process involves multiple phases, including TLS handshake, authentication, and tunnel creation.

7.4.1 Connection Establishment Process

To establish the TLS VPN connection:

```
# Navigate to client configuration directory
cd /client

# Start OpenVPN connection
openvpn --config softether.ovpn

# Alternative: Run in background
openvpn --config softether.ovpn --daemon
```

7.4.2 Connection Verification

Successful TLS tunnel establishment can be verified through several indicators:

```
# Check TUN interface creation
ip addr show | grep tun

# Test connectivity to SoftEther SecureNAT
ping 192.168.30.1

# Check assigned VPN IP address
ip addr show tun0
```

Successful Connection Indicators:

- "Initialization Sequence Completed" message appears
- TUN interface (tun0) is created with VPN IP address
- Routes are added for VPN traffic

7.5 Encrypted Traffic Analysis

TLS-based VPNs provide comprehensive encryption at the session layer, protecting all tunneled traffic.

7.5.1 Wireshark Packet Analysis

To observe TLS/OpenVPN traffic in detail, it is highly recommended to open a Wireshark instance on one of the network cables connecting the client and server infrastructure. This allows real-time analysis of the VPN establishment and data transmission phases.

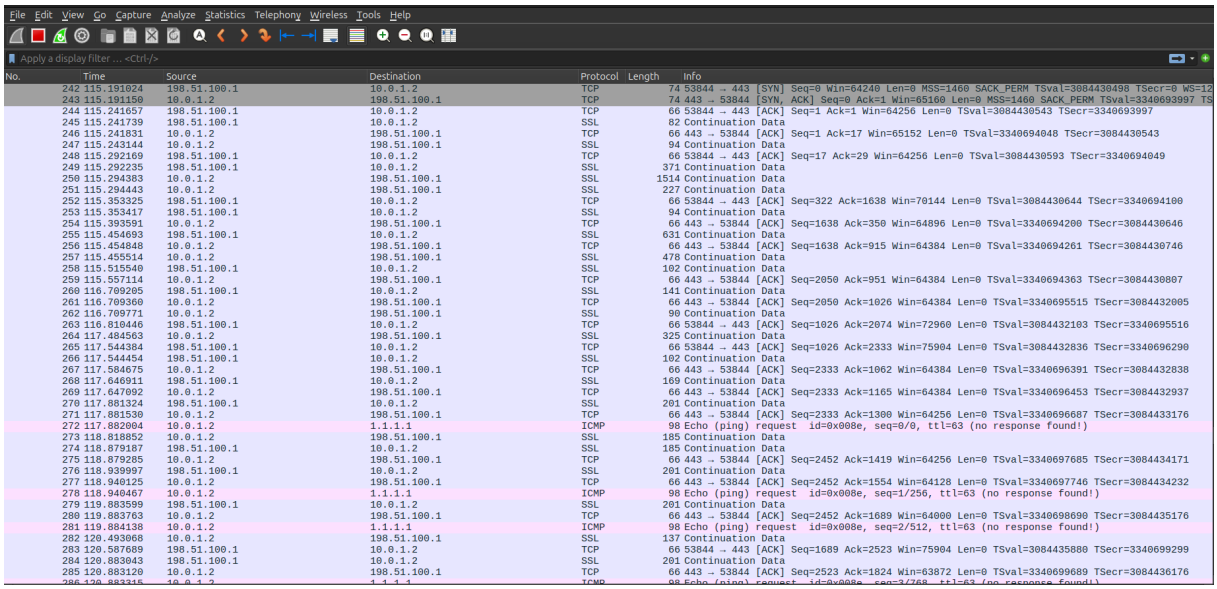

Figure 4: TSL VPN Packets in Wireshark

TLS Handshake and Connection Establishment Phase:
During the initial TLS VPN connection setup, Wireshark will capture SSL/TLS handshake packets and OpenVPN protocol messages. You can start the Wireshark capture instance by right clicking on a wire in GNS3, and then selecting "Start Capture". In this phase you will observe:

- **TCP Connection:** Initial TCP connection establishment on port 443
- **TLS Handshake packets:** SSL/TLS negotiation between client and server
- **Certificate Exchange:** Server certificate transmission and validation

Active Tunnel Communication Phase:
Once the TLS VPN tunnel is established and active communication begins, the packet capture will show encrypted SSL data packets.

Important Traffic Flow Observation:
When the client initiates communication (such as a ping command) to destinations beyond the VPN server, the ping message travels through the entire TLS tunnel to be decapsulated and transmitted by the server on behalf of the client. This means that when observing traffic with Wireshark, the source address of the ping message that appears in the packet capture will not be the original client address, but rather the SoftEther server's SecureNAT IP address.
This behavior occurs because:

- The client's traffic is completely encapsulated within the TLS tunnel
- The SoftEther server decapsulates the traffic and forwards it using its own SecureNAT functionality
- The server acts as a NAT gateway, replacing the source IP address with its own
- External destinations see traffic originating from the VPN server, not the original client

```
# Test communication to generate OpenVPN traffic and observe source translation
ping 192.168.30.1  # From client to SoftEther SecureNAT
```

7.5.2 Connection Termination

To properly terminate the VPN connection:

```
# Find OpenVPN process
ps aux | grep openvpn

# Gracefully terminate connection
kill -SIGTERM <openvpn_pid>

# Force termination if necessary
kill -SIGKILL <openvpn_pid>

# Verify interface cleanup
ip addr show | grep tun
```

The TLS-based VPN configuration is now complete and ready for testing. This implementation provides a user-space alternative to the kernel-based IPSec solution, demonstrating different approaches to achieving secure network connectivity. The next section will verify both VPN implementations and analyze their traffic characteristics.

8 Conclusion

This lab activity successfully demonstrated the implementation and comparative testing of Virtual Private Network technologies using SoftEther VPN's multi-protocol capability. Under hands-on configuration and testing, we established secure communication paths between geographically separated networks over a simulation Internet infrastructure.

8.1 Major Achievements

The lab was successful in its primary objectives:

- **Network Infrastructure:** Successfully deployed an actual GNS3 topology of three Cisco routers, Docker containers, and correct NAT configuration among them to simulate Internet-based VPN environments
- **Multi-Protocol Implementation:** Configure both IPSec (with strongSwan) and TLS/SSL (with OpenVPN) VPN connections to one single SoftEther VPN server, demonstrating the plat-form's flexibility
- **Traffic Analysis:** Used Wireshark to capture VPN setup stages and data transfer over VPN in encrypted mode, tracking ISAKMP/ESP packets for IPSec and SSL/TLS records for OpenVPN

8.2 Learning Outcomes

The lab provided experiential learning on:

- **Network Simulation:** Experiential learning in the usage of GNS3 for complex network topology configuration and management
- **VPN Technologies:** Understanding how different VPN protocols establish secure tunnels and encapsulate traffic
- **Security Analysis:** Understanding packet capture and analysis tools for VPN security property testing
- **Container Deployment:** Understanding Docker container deployment for network service deployment

8.3 Practical Insights

The lab placed focus on prime concerns of real-world VPN deployments:

- **Protocol Selection:** Whether to implement IPSec or TLS depends on factors such as existing infrastructure, client capabilities, performance requirements, and ease of installation
- **NAT Compatibility:** TLS-based solutions have better compatibility with NAT devices and restrictive firewall environments
- **Security Properties:** Both protocols implement the fundamental VPN security properties (confidentiality, integrity, authentication, anti-replay protection) but in a different manner and at a different network layer

References

[1] SoftEther VPN Project, "Softether vpn project." https://www.softether.org/. Accessed: 2024-12-18.

[2] S. Kent and K. Seo, "Security architecture for the internet protocol." RFC 4301, December 2005.

[3] N. Ferguson and B. Schneier, *A Cryptographic Evaluation of IPsec.* Counterpane Internet Security, 1999.

[4] National Institute of Standards and Technology, "Guide to ipsec vpns." https://csrc.nist.gov/publications/detail/sp/800-77/final. Special Publication 800-77, Accessed: 2024-12-18.

[5] E. Rescorla, "The transport layer security (tls) protocol version 1.3." RFC 8446, August 2018.

[6] OpenVPN Inc., "Openvpn community." https://openvpn.net/community/. Accessed: 2024-12-18.

[7] strongSwan Project, "strongswan vpn." https://www.strongswan.org/. Accessed: 2024-12-18.

[8] strongSwan Project, "strongswan documentation." https://docs.strongswan.org/. Accessed: 2024-12-18.

[9] A. Huttunen, B. Swander, V. Volpe, L. DiBurro, and M. Stenberg, "Udp encapsulation of ipsec esp packets." RFC 3948, January 2005.

[10] OpenVPN Inc., "Openvpn 2.4 manual." https://openvpn.net/community-resources/how-to/. Accessed: 2024-12-18.

[11] The OpenSSL Project, "Openssl cryptography and ssl/tls toolkit." https://www.openssl.org/. Accessed: 2024-12-18.

[12] GNS3 Technologies Inc., "Gns3 network simulator." https://www.gns3.com/. Accessed: 2024-12-18.

[13] GNS3 Technologies Inc., "Gns3 documentation." https://docs.gns3.com/. Accessed: 2024-12-18.

[14] Docker Inc., "Docker platform." https://www.docker.com/. Accessed: 2024-12-18.

[15] siomiz, "Softether vpn docker image." https://hub.docker.com/r/siomiz/softethervpn. Accessed: 2024-12-18.

[16] Canonical Ltd., "Ubuntu official docker images." https://hub.docker.com/_/ubuntu. Accessed: 2024-12-18.