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 Laboratory Activity

License

Acknowledgments

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

Contents

1 Introduction

This laboratory activity focuses on the implementation and analysis of **Virtual Private Networks** (VPNs) using SoftEther VPN technology across an Internet infrastructure. VPNs are fundamental network security solutions that provide secure connectivity over 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 distributed networks, simulating real-world scenarios where branch offices or remote users need secure access to corporate resources. Through practical implementation, we will explore two distinct VPN protocols: IPSec and TLS/SSL, analyzing their different approaches to achieving network security.

1.1 Laboratory Objectives

The primary goals of this laboratory activity include:

- **Understanding VPN Fundamentals:** Explore the theoretical foundations of Virtual Private Networks and their role in modern network security architectures.
- **SoftEther VPN Implementation:** Configure and deploy SoftEther VPN server to provide multi-protocol VPN services, as IPSec and TLS-based connections.
- **Network Topology Design:** Implement a realistic network topology using GNS3 that simulates Internet connectivity between two remote sites through an ISP infrastructure.
- **Protocol Comparison:** Analyze and compare different VPN protocols (IPSec vs. TLS) in terms of security properties, performance characteristics, and implementation complexity.
- **Traffic Analysis:** Perform detailed packet analysis using Wireshark to understand how different VPN protocols encapsulate and protect network traffic.
- **Security Assessment:** Evaluate the security features provided by each VPN implementation, including encryption, authentication, and integrity protection.

1.2 SoftEther VPN Overview

SoftEther VPN is a multi-protocol VPN software that supports various VPN protocols including IPSec, L2TP, OpenVPN (TLS/SSL), and proprietary protocols. Developed by the University of Tsukuba, it provides a unified platform for implementing different VPN technologies, making it an excellent choice for educational and research purposes.
The key advantages of SoftEther VPN include:

- Multi-protocol support enabling comparison of different VPN approaches
- Cross-platform compatibility (Windows, Linux, macOS)
- High performance with optimized networking code
- Comprehensive logging and monitoring capabilities
- Support for complex network topologies and routing scenarios

1.3 Laboratory Environment and Network Topology

This laboratory simulates a scenario where two geographically separated private networks need to establish secure communication across 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 capabilities
- **Internet Infrastructure:** Simulated ISP network providing connectivity between sites

- **VPN Gateways:** Edge routers performing NAT and routing functions, to which a public IP address is assigned

The implementation demonstrates how VPN technology enables secure site-to-site connectivity, allowing hosts in different private networks to communicate as if they were on the same local network, while maintaining security properties across the untrusted Internet infrastructure.

1.4 System Requirements

To successfully complete this laboratory 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:** Network simulation platform
- **Docker:** Container platform for VPN endpoints
- **Wireshark:** Network protocol analyzer

Hardware Requirements:

- Minimum 8GB RAM (16GB recommended for optimal performance)
- 20GB available disk space
- Network interface with Internet connectivity

Software Installation:
The required software can be installed on Ubuntu/Debian systems using the following commands:

```
# Update package repositories
sudo apt update

# 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: After installation, a logout and login cycle may be required for group membership changes to take effect. The VPN client software (strongSwan for IPSec and OpenVPN for TLS) will be installed inside the Docker containers as part of the laboratory setup.

1.5 Laboratory Structure

This report is organized into the following sections:

1. **Theoretical Background:** Comprehensive overview of VPN technologies, SoftEther VPN architecture, and the security protocols involved.

2. **Network Topology:** Detailed description of the simulated network infrastructure and addressing scheme.
3. **Initial Configuration:** Step-by-step setup of the GNS3 environment, routers, and container infrastructure.
4. **SoftEther VPN Server:** Configuration and deployment of the VPN server with multi-protocol support.
5. **IPSec Configuration:** Implementation of IPSec-based VPN using strongSwan.
6. **TLS Configuration:** Setup of TLS/SSL VPN using OpenVPN client.

Each section includes practical implementation steps, configuration examples, and analysis of the results, providing a comprehensive understanding of VPN technologies and their real-world applications in network security.

2 Theoretical Background

This section provides the theoretical foundation necessary to understand the Virtual Private Network technologies implemented in this laboratory. We examine the fundamental concepts of VPNs, the specific architecture of SoftEther VPN, and the security protocols that enable secure communication across untrusted networks.

2.1 Virtual Private Networks (VPNs)

Virtual Private Networks are network security solutions that provide connectivity over shared infrastructure while maintaining properties typically associated with 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 several essential properties to ensure secure communication:

- **Data Confidentiality:** Encryption algorithms protect data content from unauthorized access during transmission across public networks.
- **Data Integrity:** Cryptographic hash functions and message authentication codes ensure that data has not been modified in transit.
- **Authentication:** Both peer authentication (verifying the identity of communicating endpoints) and data origin authentication are essential for secure VPN operation.
- **Access Control:** VPNs implement policies that determine which traffic is permitted and how it should be processed.
- **Anti-Replay Protection:** Sequence numbers and timestamps prevent malicious replay of previously captured packets.

VPN Classification:
VPNs can be classified based on their implementation approach and the network layers where security is enforced. The layers analyzed in this laboratory are:

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

2.2 SoftEther VPN Architecture

SoftEther VPN represents a unique approach to VPN implementation by providing a unified platform that supports multiple VPN protocols simultaneously. This multi-protocol capability enables direct comparison and evaluation of different VPN technologies within a single deployment.

Multi-Protocol Support:
SoftEther VPN's architecture is designed around the concept of *protocol abstraction*, where different VPN protocols are implemented as modules within a common framework. The main protocols that we will use in this laboratory are:

- **IPSec Support:** Native implementation of 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 that can:

- Aggregate multiple VPN sessions from different protocols

- Implement advanced routing and bridging capabilities
- Provide centralized policy enforcement and logging
- Support complex network topologies including site-to-site and remote access scenarios

SecureNAT Functionality:
SoftEther VPN includes an integrated NAT (Network Address Translation) and DHCP server functionality called SecureNAT, which simplifies VPN client configuration by:

- Automatically assigning IP addresses to VPN clients
- Providing DNS resolution services
- Enabling Internet access for VPN clients through NAT
- Supporting both IPv4 and IPv6 addressing schemes

2.3 IPSec Protocol Suite

Internet Protocol Security (**IPSec**) is a framework providing security at the IP layer by adding headers to IP packets. It can operate in transport mode (protecting payload only) or tunnel mode (encapsulating the entire packet).
One key component is the **Encapsulating Security Payload** (ESP), which provides confidentiality through encryption and may also include integrity and authentication. 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 such as cryptographic algorithms, keys, and endpoints; each SA is identified by a Security Parameter Index (SPI). Automated management of these SAs is typically handled by Internet Key Exchange protocols: **IKEv1** (with main and aggressive modes) and **IKEv2** (which improves performance and security). In this project, IPSec (via ESP, SA, and IKE) is used to establish a secure tunnel at the network layer for scenarios requiring kernel-level packet protection.

2.4 TLS/SSL for VPN Implementation

Transport Layer Security (TLS) provides session-layer security by performing a handshake to negotiate cipher suites, authenticate the server (and optionally the client), exchange keys—often providing perfect forward secrecy—and then protecting application data in a record layer. TLS-based VPNs (such as **OpenVPN**) encapsulate arbitrary IP traffic within a TLS channel, leveraging TLS's widespread support and flexible cipher negotiation. This approach places security processing in user space, which can be easier to deploy and customize, though with some performance trade-offs compared to kernel-based solutions.

2.5 Client-Side VPN Technologies

StrongSwan (IPSec):
strongSwan is an open-source IPSec implementation used in this project to establish and manage IPSec tunnels. It supports IKEv1 and IKEv2, a range of cryptographic algorithms, and features like NAT traversal and Dead Peer Detection. Configuration is driven by simple files (e.g., `ipsec.conf` and `ipsec.secrets`), and it integrates with the Linux kernel's IPSec stack. In our setup, strongSwan handles automated key exchanges and maintains SAs for secure network-layer tunnels.

OpenVPN Client (TLS):
OpenVPN is the TLS-based client solution in this project. It uses OpenSSL for cryptography, handles both UDP and TCP transports, and can operate in tun (Layer 3) or tap (Layer 2) mode. Certificate-based authentication via a certificate ensures that only authorized endpoints connect. OpenVPN's user-space implementation allows flexible scripting hooks and logging, making it suitable for custom remote-access scenarios in the laboratory.

2.6 Comparative Analysis Framework

Understanding the theoretical differences between IPSec and TLS implementations lays the groundwork for practical comparison:

- **Architecture Differences:** IPSec is implemented at the network layer (often in kernel space), whereas TLS-based VPNs run in user space at the session layer.
- **Performance Considerations:** IPSec generally has lower overhead due to kernel integration; TLS-based VPNs offer more flexibility but may incur extra processing in user space.
- **Deployment Complexity:** IPSec setups can be more complex but follow well-established standards, while TLS-based VPNs like OpenVPN often allow simpler initial configuration and greater customization.

This theoretical foundation establishes the necessary knowledge base for understanding the practical implementation and comparison of SoftEther VPN's multi-protocol capabilities in the subsequent sections of this laboratory activity.

3 Network Topology and Architecture

This section presents the network topology designed for the SoftEther VPN laboratory activity. The implementation simulates a realistic scenario where two geographically separated private networks communicate securely across the Internet through VPN tunnels.

3.1 Topology Overview

The laboratory network topology consists of five main components that work together to create a realistic Internet-based VPN scenario:

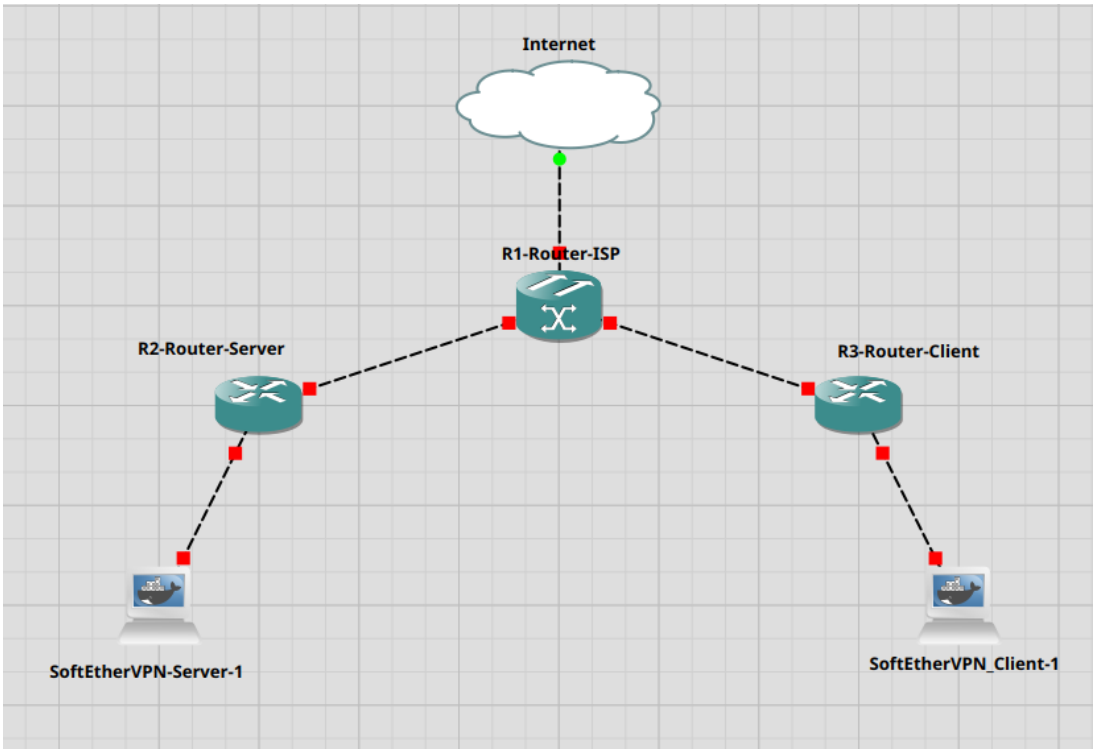

Figure 1: GNS3 Network Topology - VPN Across the Internet

The network architecture follows an end-to-end VPN design where:

- **Server Site:** Contains the SoftEther VPN server hosted within a private network
- **Client Site:** Represents a remote branch office requiring VPN connectivity
- **Internet Infrastructure:** Simulated 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 effectively demonstrates how end-to-end VPN technology enables secure communication between hosts in different private networks across untrusted public infrastructure, representing common real-world deployment scenarios where VPN software runs directly on the endpoints rather than on gateway devices.

3.2 IP Addressing Scheme

The network addressing scheme uses a combination of private and public IP addresses to simulate realistic Internet connectivity. The addressing follows RFC standards for private addressing (RFC 1918) and uses documentation addresses (RFC 5737) for public IP simulation.

Table 1: Network Interface Configuration

Device	Interface	IP Address	Description
Router 2 (Server-side)	LAN (Fa0/0) WAN (Fa0/1)	10.0.1.1/24 203.0.113.1/24	Private Network 1 Public IP (RFC 5737)
ISP Router	Interface Fa0/0 Interface Fa0/1 Interface Fa1/0	203.0.113.254/24 198.51.100.254/24 DHCP Assigned	Connected to Router 2 Connected to Router 3 Internet Cloud
Router 3 (Client-side)	WAN (Fa0/1) LAN (Fa0/0)	198.51.100.1/24 10.0.2.1/24	Public IP (RFC 5737) 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 hosting the SoftEther VPN server
- **Client Network (10.0.2.0/24):** Remote private subnet with 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 for VPN connectivity (assigned dynamically)

3.3 Device Roles and Functions

Each component in the network topology serves a specific purpose in demonstrating VPN functionality:

3.3.1 ISP Router (R1-Router-ISP)

The ISP router simulates Internet Service Provider infrastructure and provides:

- **Internet Connectivity:** DHCP-assigned public IP address with MAC address spoofing for network access (Explained later).
- **Inter-Site Routing:** Routes traffic between the two remote sites through simulated Internet infrastructure
- **NAT Traversal Support:** Enables VPN protocols to function through Network Address Translation
- **Public IP Assignment:** Provides public addressing for both edge routers

The ISP router configuration includes static routes for both private networks and implements a default route to the Internet cloud.

3.3.2 Server-Side Router (R2-Router-Server)

This edge router connects the server's private network to the Internet and implements:

- **Network Address Translation:** Port Address Translation management
- **VPN Port Forwarding:** Static NAT rules to forward 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

3.3.3 Client-Side Router (R3-Router-Client)

The client-side edge router provides similar functionality for the remote site:

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

3.3.4 SoftEther VPN Server Container

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

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

3.3.5 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

3.4 Network Communication Flow

The network design enables several 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 initiates 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 VPN tunnels are established, the client can communicate securely with resources in the server's network
4. **NAT Traversal:** VPN protocols handle NAT traversal automatically, allowing encrypted tunnels to pass through the edge routers

3.5 Security Considerations

The network topology incorporates several security features:

- **Private Addressing:** Internal networks use RFC 1918 private addresses, preventing direct Internet access
- **NAT Protection:** Edge routers provide implicit firewall protection through NAT
- **VPN Encryption:** All inter-site communication is protected by VPN encryption
- **Authentication:** VPN connections require user credentials and/or certificates
- **Protocol Separation:** Different VPN protocols operate on distinct ports for protocol isolation

This topology provides a foundation for comparing different VPN implementations while maintaining realistic network security practices and demonstrating practical deployment scenarios.

4 Initial Network Configuration

This section details the step-by-step process of setting up the laboratory environment, including GNS3 configuration, router setup, and Docker container deployment. The configuration establishes the basic network infrastructure that will support the VPN implementations.

4.1 GNS3 Environment Setup

The laboratory utilizes GNS3 (Graphical Network Simulator-3) to create a realistic network topology. GNS3 provides the platform for simulating Cisco routers, Docker containers, and network connectivity required for the VPN demonstration.

4.1.1 Device Import and Template Creation

Cisco Router Template:
The topology requires Cisco 7200 routers running IOS version 124-24.T5. This specific IOS image is chosen because it represents a freely available alternative to modern commercial Cisco IOS versions. While newer platforms like IOSv (IOS Virtual) offer enhanced features and performance, they require valid Cisco licensing agreements which can be costly for educational purposes. The 124-24.T5 image, although older and not widely deployed in current production environments, provides all the fundamental routing and NAT capabilities required for this laboratory 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. Import the router image file to complete the setup

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

- **Server Container Image:** `siomiz/softethervpn:latest`
- **Client Container Image:** `ubuntu:latest`

To add containers to the 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. Repeat for the Ubuntu client container

You will now see the containers in the left panel of GNS3, ready to be dragged into the workspace. So now you can recreate the same topology as shown in the figure in the section 3 of this document. Clicking on the wire on the left panel, you will be able to connect the devices together, creating the network topology defined. Choose the right interfaces for each connection as shown in the table in the section 3.2.

Important 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 necessary because the ISP router needs to connect to both edge routers and the Internet cloud. To configure the ISP router slots:

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 require specific configuration for persistent storage and network setup.

4.2.1 Server Container Setup

The SoftEther VPN server container needs persistent directories for configuration files:

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

This directory will be created in the GNS3 project folder, allowing persistent storage of configuration files between container restarts.

Server Configuration File Setup:
To ensure the SoftEther VPN server starts with the proper configuration, you must place the `vpn_server.config` file in the correct 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 server container folder (usually named after the container)
5. Navigate to the `usr/vpnserver` directory within the container folder
6. Place the `vpn_server.config` file in this directory

This ensures that when the container starts, the SoftEther VPN service will automatically load the pre-configured settings, including user accounts, IPSec settings, and SSL/TLS configurations.

4.2.2 Client Container Setup

The client container requires similar configuration 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 multiple configuration files for both 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 necessary VPN client configuration files are available when the container starts, allowing seamless connection to both IPSec and TLS VPN services.

4.3 Project Startup and Device Access

Once all configurations are complete and files are properly placed in their persistent directories, you can start the laboratory environment.

4.3.1 Starting the GNS3 Project

To start all devices and containers in the project:

1. Click the green **Play button** in the top toolbar of GNS3
2. This will start all routers and Docker containers simultaneously
3. Wait for all devices to complete their boot process
4. Verify that all device icons show a green status indicator

4.3.2 Accessing Device Terminals

To interact with the network devices and containers:
For Cisco Routers:

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

For Docker Containers:

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

Important Note: For Docker containers, always use the "Auxiliary Console" option rather than the regular console, as it provides the proper terminal interface for interacting with the containerized operating system.

4.4 Router Configuration

Now that you can access the router terminals, proceed with configuring the three routers with specific configurations to simulate 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 configuration 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:
The MAC address spoofing is necessary because the Politecnico di Torino WiFi network's DHCP server implements MAC address filtering for security purposes. Without using the host machine's MAC address, the DHCP server will not assign a valid IP address to the router interface, preventing Internet connectivity for the simulated network. To obtain 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:** 1c:ce:51:3f:98:20
- **Cisco format:** 1cce.513f.9820

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)'

# Expected output should show:
# tcp LISTEN 0.0.0.0:443
# udp LISTEN 0.0.0.0:500
# udp LISTEN 0.0.0.0:4500
```

4.7 Network Infrastructure Validation

At this point, the basic network infrastructure should be operational 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
- Basic inter-site connectivity through the simulated Internet

This foundation enables the implementation of VPN services detailed in the subsequent sections. Any connectivity issues at this stage should be resolved before proceeding with VPN configuration, as they will prevent proper VPN tunnel establishment.
The next section will detail the configuration of the SoftEther VPN server to provide multi-protocol VPN services for both IPSec and TLS-based connections.

5 SoftEther VPN Server Configuration

This section details the configuration and deployment of the SoftEther VPN server within the Docker container environment. The server provides multi-protocol VPN services, supporting both IPSec and TLS/SSL connections simultaneously through a unified management interface.

5.1 Server Installation and Initialization

The SoftEther VPN server runs within a Docker container using the pre-built image `siomiz/softethervpn:latest`. This containerized approach provides several advantages including isolation, portability, and simplified deployment.

5.1.1 Container Deployment

The server container has been configured in GNS3 with persistent storage directories as detailed in Section 4. When the container starts, the SoftEther VPN service initializes automatically with the following characteristics:

- **Automatic Service Start:** The VPN server daemon starts automatically when the container boots, thanks to the persistent storage configuration, the server will start already with the proper configuration file.
- **Multi-Protocol Listeners:** Simultaneous support for IPSec, SSL/TLS
- **Network Integration:** Automatic integration with the container's network interface (eth0)

5.1.2 Service Verification

To verify the SoftEther VPN server is running correctly, execute the following commands 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 managed through the `vpn_server.config` file, which contains comprehensive settings for all VPN protocols, virtual hubs, user management, and security policies.

5.2.1 Core Server Settings

The server configuration includes several critical components:

```
# 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 major VPN protocols are enabled by default
- **NAT Traversal:** Essential for clients behind NAT devices
- **Security Settings:** Strong cipher suites with DHE for perfect forward secrecy
- **Connection Limits:** Reasonable limits to prevent resource exhaustion
- **Logging:** Comprehensive logging for troubleshooting and analysis

5.2.2 Protocol Listener Configuration

The server configures multiple listeners for different 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
    }
}
```

5.3 Virtual Hub and User Management

SoftEther VPN organizes connections through Virtual Hubs, which act as virtual Ethernet switches. The default configuration 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 certificate authentication for the mutual authentication because of the lab environment)
- **SecureNAT:** Integrated DHCP and NAT services for client IP assignment
- **Access Control:** Configurable policies for traffic filtering and routing
- **Logging:** Comprehensive logging of user sessions and traffic

5.3.2 User Account Management

The server 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 $
    }
}
```

5.4 IPSec Protocol Configuration

The SoftEther VPN server includes native IPSec support, allowing standard IPSec clients to connect without additional software. The pre-shared key is not a strong one here, but it is suitable for laboratory use.

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
- **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
- **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 laboratory use
- **Usage:** Server authentication for TLS/SSL connections

5.6.2 Security Policies

The server implements several security policies:

```
# Security-related settings
# Enable DoS protection
bool DisableDosProction 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

After configuration, verify the server is operating correctly and ready to accept VPN connections.

5.7.1 Service Status Check

```
# Verify all VPN protocols are listening
netstat -tuln | grep -E '(443|500|4500|992|5555)'

# Test connectivity from client network
# From client container:
ping 203.0.113.1    # Should reach server's public IP
telnet 203.0.113.1 443  # Should connect to HTTPS port
```

5.7.2 Configuration Validation

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 and 992 are 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 now configured and ready to accept connections from both IPSec and SSL/TLS VPN clients. The next sections will detail the configuration of these client types and demonstrate secure tunnel establishment across the simulated 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

strongSwan is a complete IPSec implementation for Linux systems that supports both IKEv1 and IKEv2 protocols. It integrates with the Linux kernel's IPSec stack to provide high-performance encrypted tunnels.

6.1.1 Software Installation

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
ipsec version
```

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
- **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". In this phase you will observe:

- **ISAKMP packets on UDP port 500:** Initial key exchange and authentication
- **NAT-T packets on UDP port 4500:** NAT traversal negotiation if NAT is detected
- **Phase 1 and Phase 2 negotiations:** Security Association establishment

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

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 validation ensures server identity
2. **Certificate Name Verification:** CN matching prevents man-in-the-middle attacks
3. **User Authentication:** Username/password provides additional access control
4. **Strong Encryption:** AES-128-CBC provides confidentiality
5. **Transport Security:** TLS transport layer provides additional protection

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
```

```
# Check connection status
ps aux | grep openvpn
```

7.4.2 Connection Verification

Successful TLS tunnel establishment can be verified through several indicators:

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

# Verify VPN routes
ip route 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
- Connectivity to SecureNAT gateway is established

7.4.3 TLS Handshake Analysis

The TLS handshake process includes several critical steps:

1. **TCP Connection:** Establish TCP connection to server port 443
2. **TLS Handshake:** Negotiate cipher suites and exchange certificates
3. **Server Authentication:** Verify server certificate against CA
4. **Key Exchange:** Establish session keys for encryption
5. **User Authentication:** Submit username/password credentials
6. **Tunnel Establishment:** Create TUN interface and configure routing

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.
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

- **OpenVPN Control Messages:** Protocol-specific negotiation and authentication

Active Tunnel Communication Phase:
Once the TLS VPN tunnel is established and active communication begins, the packet capture will show encrypted OpenVPN data packets. In the case of TLS VPN, the tunnel between the hosts works in a specific way that demonstrates the complete end-to-end functionality.

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 (10.0.2.2), but rather the SoftEther server's SecureNAT IP address (192.168.30.1).
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
# The ping will appear in Wireshark with source IP 192.168.30.1, not 10.0.2.2
```

During active communication, you will observe:

- **OpenVPN Data Packets:** Encrypted application data within TLS records
- **SSL/TLS Records:** All traffic encapsulated within TLS protocol
- **Encrypted Payload:** Data content completely protected by TLS encryption
- **TCP Reliability:** Reliable delivery through TCP transport protocol
- **Source Address Translation:** Traffic appears to originate from the VPN server's SecureNAT IP

7.5.2 Encryption Characteristics

The TLS implementation provides several security properties:

- **Confidentiality:** AES-128-CBC encryption protects data content
- **Integrity:** HMAC ensures data has not been modified
- **Authentication:** Certificates and credentials verify endpoint identity
- **Perfect Forward Secrecy:** Session keys are independent of long-term keys
- **Replay Protection:** Sequence numbers prevent packet replay attacks

7.5.3 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 laboratory activity successfully demonstrated the implementation and comparative analysis of Virtual Private Network technologies using SoftEther VPN's multi-protocol capabilities. Through practical configuration and testing, we established secure communication channels between geographically separated networks across a simulated Internet infrastructure.

8.1 Key Achievements

The laboratory accomplished its primary objectives by:

- **Network Infrastructure:** Successfully implemented a realistic GNS3 topology with three Cisco routers, Docker containers, and proper NAT configuration to simulate Internet-based VPN scenarios
- **Multi-Protocol Implementation:** Configured both IPSec (using strongSwan) and TLS/SSL (using OpenVPN) VPN connections to a single SoftEther VPN server, demonstrating the platform's versatility
- **Traffic Analysis:** Used Wireshark to analyze VPN establishment phases and encrypted data transmission, observing ISAKMP/ESP packets for IPSec and SSL/TLS records for OpenVPN

8.2 Protocol Comparison

The comparative analysis revealed distinct characteristics of each VPN approach:

IPSec VPN:

- Operates at the network layer with kernel-level integration
- Provides efficient performance but complex configuration
- Requires NAT-T for operation behind NAT devices
- Uses pre-shared key authentication in our implementation

TLS/SSL VPN:

- Operates in user space with simpler deployment
- Easily traverses firewalls using standard TCP port 443
- Provides certificate-based authentication with X.509 PKI
- Offers greater configuration flexibility but higher CPU overhead

8.3 Learning Outcomes

This laboratory provided hands-on experience with:

- **Network Simulation:** Practical skills in using GNS3 for complex network topology creation and management
- **VPN Technologies:** Deep understanding of how different VPN protocols establish secure tunnels and handle traffic encapsulation
- **Security Analysis:** Experience with packet capture and analysis techniques for evaluating VPN security properties
- **Container Deployment:** Knowledge of Docker container configuration for network service deployment

8.4 Practical Insights

The laboratory highlighted important considerations for real-world VPN deployments:

- **Protocol Selection:** The choice between IPSec and TLS depends on factors such as existing infrastructure, client capabilities, performance requirements, and ease of deployment
- **NAT Compatibility:** TLS-based solutions generally provide better compatibility with NAT devices and restrictive firewall environments
- **Security Properties:** Both protocols provide essential VPN security features (confidentiality, integrity, authentication, anti-replay protection) but through different mechanisms and at different network layers

8.5 Final Remarks

The SoftEther VPN laboratory successfully demonstrated the practical implementation of multiple VPN technologies within a unified platform. The experience provided valuable insights into VPN protocol differences, network security implementation, and the importance of selecting appropriate technologies based on specific deployment requirements.

The multi-protocol capability of SoftEther VPN proved excellent for educational purposes, enabling direct comparison of different VPN approaches. This laboratory reinforced the critical role of VPN technologies in modern network security and provided practical skills necessary for implementing secure communication solutions in real-world environments.

The knowledge gained through this activity establishes a solid foundation for future work in network security, demonstrating that properly implemented VPN technologies provide robust solutions for protecting network communications across untrusted infrastructure.