

Project Report: Security Analysis and Implementation of IPSec Site-to-Site Tunnel

Course Name: Network Management

Project Title: Security Analysis and Implementation of IPSec Site-to-Site Tunnel

Asal Dehghan
Hooman Ebrahimi
Artin Kiaee

1. Introduction

1.1 Project Goal

The primary objective of this project was to design, implement, and analyze a secure site-to-site Virtual Private Network (VPN) tunnel using the IPSec protocol. This tunnel establishes a secure communication channel between two distinct network sites, simulating a Headquarters (HQ) and a Branch Office (BR), with the aim of protecting sensitive data transmitted over an untrusted network.

1.2 Importance of IPSec

IPSec (Internet Protocol Security) is a suite of protocols that provides cryptographic security services at the IP layer. It ensures data confidentiality (encryption), integrity (data authentication), and authenticity (source authentication) for IP packets. Operating at Layer 3 of the OSI model, IPSec is a fundamental technology for securing communications over public networks like the internet, making it indispensable for secure remote access, site-to-site connectivity, and e-commerce.

1.3 Report Overview

This report details the comprehensive process undertaken to implement the IPSec tunnel. It covers the network design, virtual machine setup, strongSwan IPSec configuration, verification steps, and a preliminary analysis of the tunnel's security and performance. Key troubleshooting steps encountered during the implementation are also documented to provide a complete picture of the project's execution.

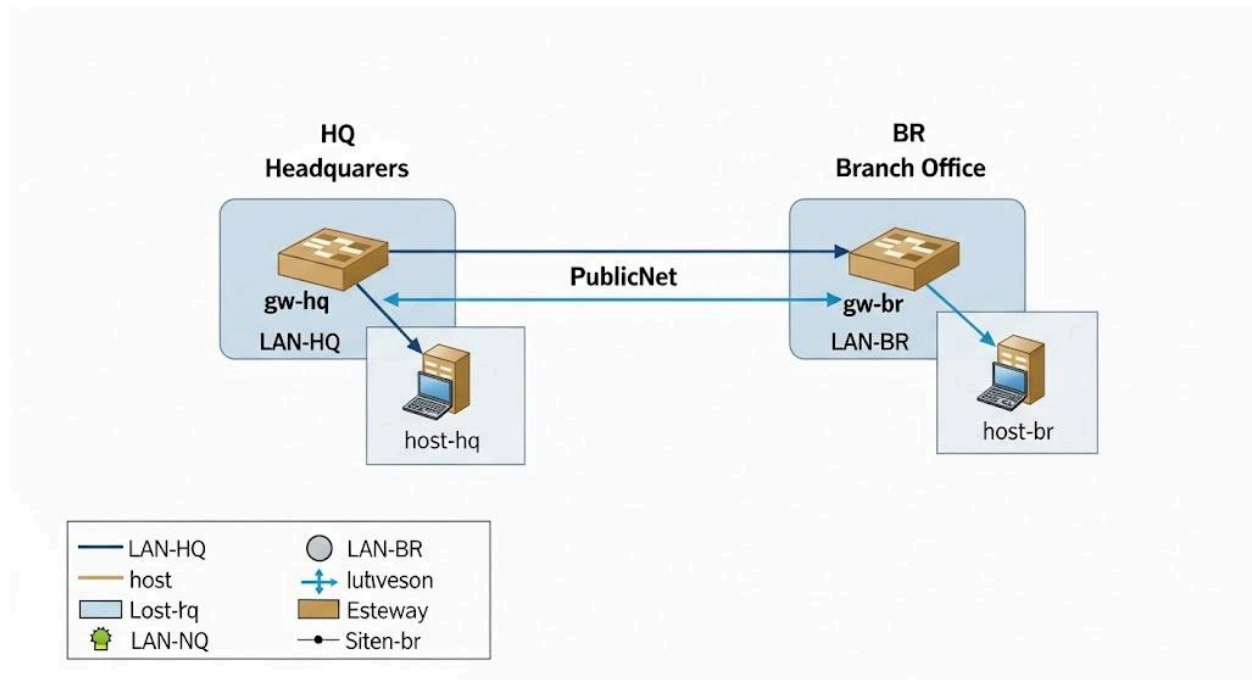
2. Network Design and Virtualization Setup

To simulate a realistic network environment for the IPSec tunnel, VirtualBox was chosen as the virtualization platform. The design incorporates two distinct local area networks (LANs) connected via gateway machines over a simulated public network.

2.1 Network Topology

The network topology comprises two main sites: Headquarters (HQ) and Branch Office (BR). Each site consists of a gateway machine and a host machine. These sites are interconnected via a simulated "PublicNet" (representing the internet).

- **Headquarters (HQ) Network:** 10.0.0.0/24
 - gw-hq: HQ Gateway (IPSec endpoint)
 - host-hq: HQ Local Host
- **Branch Office (BR) Network:** 10.1.0.0/24
 - gw-br: BR Gateway (IPSec endpoint)
 - host-br: BR Local Host
- **Public Network:** 192.168.99.0/24 (VirtualBox NAT Network)



2.2 VirtualBox Network Configuration

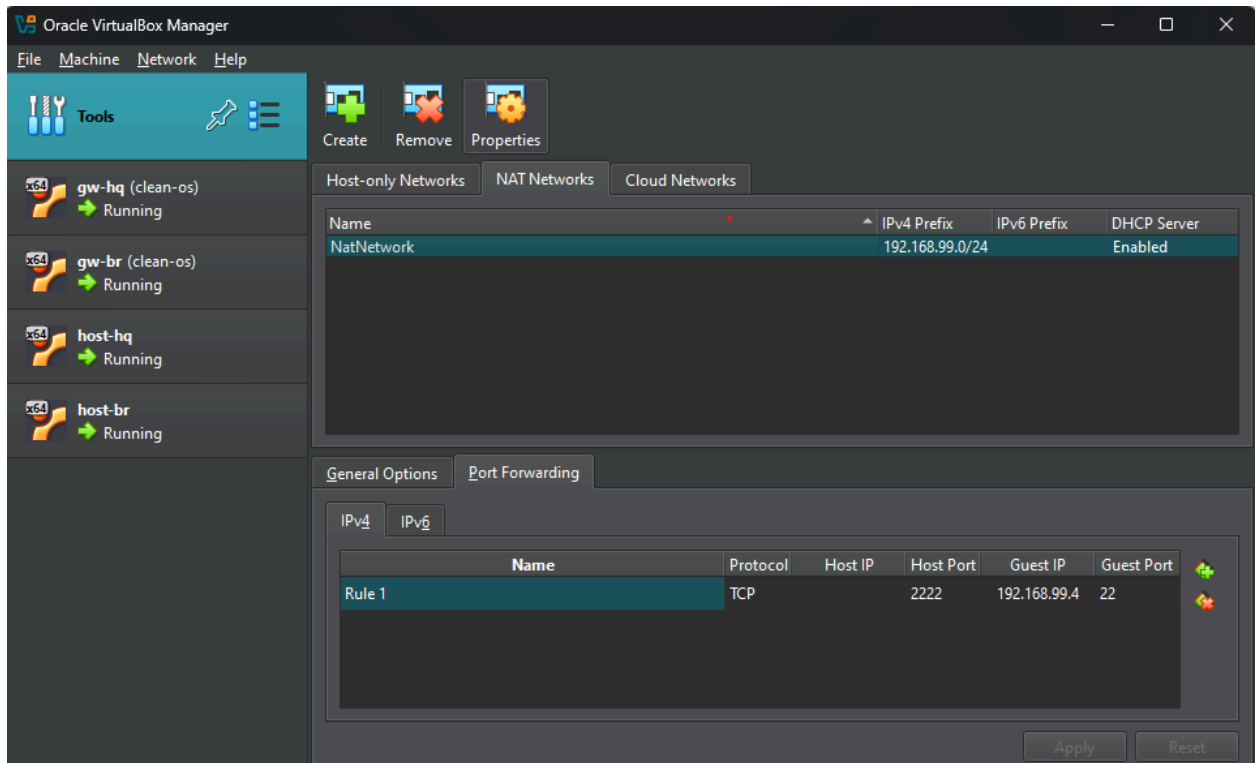
The following network adapters were configured within VirtualBox to establish the desired topology:

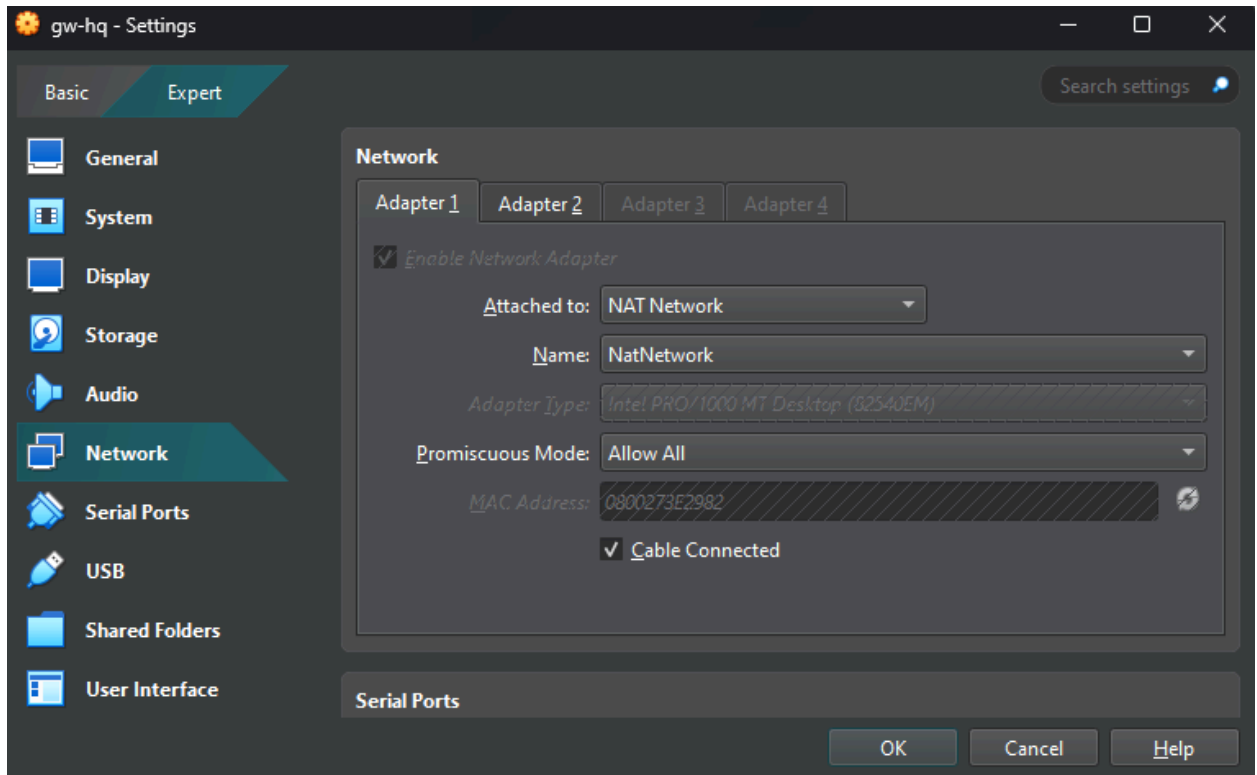
- NAT Network – "PublicNet"

This network simulates the public internet, allowing the gateway VMs to communicate with each other using routable "public" IP addresses. DHCP was enabled to automatically assign IPs to the gateway's public-facing interfaces.

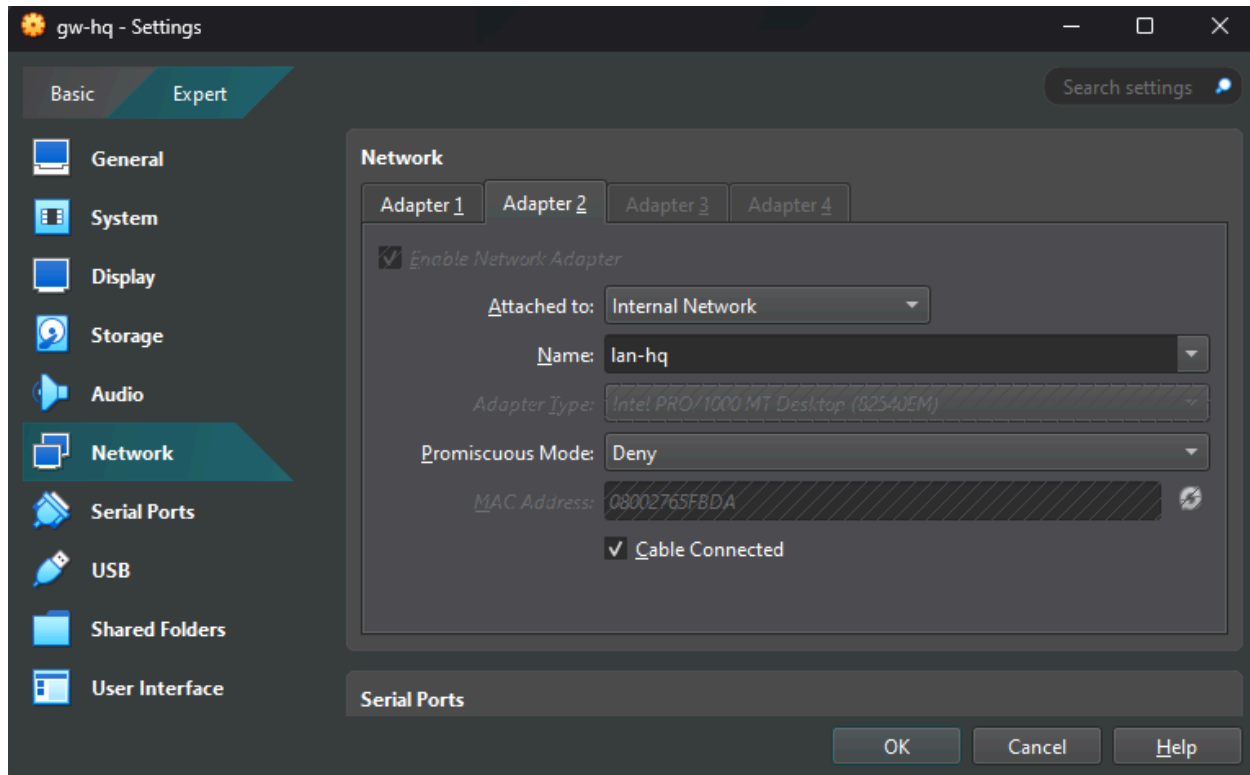
 - **Configuration Steps:**
 1. VirtualBox GUI.
 2. File ► Preferences ► Network ► NAT Networks.

3. adding a new NAT Network.
4. **Name:** NatNetwork
5. **Network CIDR:** 192.168.99.0/24
6. **Enable DHCP** is checked.





- Internal Networks – "LAN-HQ" and "LAN-BR"
These networks provide isolated local area networks for the HQ and Branch Office, ensuring that traffic within each LAN is separate and only routed through their respective gateways. Internal networks are created by simply referencing their names when attaching a VM's network adapter.



2.3 Virtual Machine Setup

Four primary virtual machines were created and configured with Ubuntu Server (for gateways) and Ubuntu Desktop/Server (for hosts) operating systems.

- Gateway VM #1 — gw-hq
 - This VM acts as the IPsec endpoint for the HQ network.
 - **VirtualBox GUI Setup:**
 1. New ► **Name:** gw-hq, **Type:** Linux, **Version:** Ubuntu (64-bit).
 2. **Memory:** 2048 MB, **Disk:** 10 GB VDI (dynamically allocated).
 3. **Settings ► Network:**
 - **Adapter 1:** NAT Network, **Name:** PublicNet, **Promiscuous Mode:** Allow VMs.
 - **Adapter 2:** Internal Network, **Name:** LAN-HQ.
 4. **Settings ► Storage:** Attach the Ubuntu Server ISO to the optical drive.
 - Ubuntu Installation (Netplan Configuration):

During the Ubuntu Server installation, network interfaces were configured as follows:

 - **Adapter 1 (e.g., ens3):** DHCP (to get an IP from PublicNet, e.g., 192.168.99.4)
 - **Adapter 2 (e.g., ens4):** Static, **IP:** 10.0.0.1/24, **Gateway:** — (blank), **DNS:** 8.8.8.8.

- After installation, the ISO was ejected, and a snapshot named "clean-OS" was taken.

```
root@gw-hq:/home/art# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:3e:29:82 brd ff:ff:ff:ff:ff:ff
    inet 192.168.99.4/24 metric 100 brd 192.168.99.255 scope global dynamic enp0s3
        valid_lft 572sec preferred_lft 572sec
    inet6 fe80::a00:27ff:fe3e:2982/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:65:fb:da brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.1/24 brd 10.0.0.255 scope global enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe65:fbda/64 scope link
        valid_lft forever preferred_lft forever
```

- Gateway VM #2 — gw-br

This VM acts as the IPSec endpoint for the Branch Office network.

- **VirtualBox GUI Setup:**

1. Repeat steps for gw-hq, but with **Name:** gw-br.
2. **Settings ► Network:**
 - **Adapter 1:** NAT Network, **Name:** PublicNet.
 - **Adapter 2:** Internal Network, **Name:** LAN-BR.

- **Ubuntu Installation (Netplan Configuration):**

- **Adapter 1 (e.g., ens3):** DHCP (to get an IP from PublicNet, e.g., 192.168.99.5)
- **Adapter 2 (e.g., ens4):** Static, **IP:** 10.1.0.1/24, **Gateway:** — (blank), **DNS:** 8.8.8.8.

```
^Cart@gw-br:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:95:4b:d6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.99.5/24 metric 100 brd 192.168.99.255 scope global dynamic enp0s3
        valid_lft 378sec preferred_lft 378sec
    inet6 fe80::a00:27ff:fe95:4bd6/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:bb:04:aa brd ff:ff:ff:ff:ff:ff
    inet 10.1.0.1/24 brd 10.1.0.255 scope global enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:febb:4aa/64 scope link
        valid_lft forever preferred_lft forever
```

- LAN Host VMs (host-hq and host-br)

These VMs represent typical hosts within each local network.

- **host-hq:**
 - **Name:** host-hq, **Memory:** 2048 MB.
 - **Network:** Internal Network, **Name:** LAN-HQ (only Adapter 1).
 - **IP Configuration:** Static, **IP:** 10.0.0.10/24, **Gateway:** 10.0.0.1.
- **host-br:**
 - **Name:** host-br, **Memory:** 2048 MB.
 - **Network:** Internal Network, **Name:** LAN-BR (only Adapter 1).
 - **IP Configuration:** Static, **IP:** 10.1.0.10/24, **Gateway:** 10.1.0.1.

3. IPSec Tunnel Implementation with strongSwan

With the virtual network infrastructure in place, the strongSwan IPSec daemon was configured on both gateway machines (gw-hq and gw-br) to establish the secure tunnel.

3.1 strongSwan Installation

strongSwan and its utilities were installed on both gw-hq and gw-br:

```
sudo apt update
sudo apt install strongswan strongswan-swanctl
```

3.2 Enable IP Forwarding

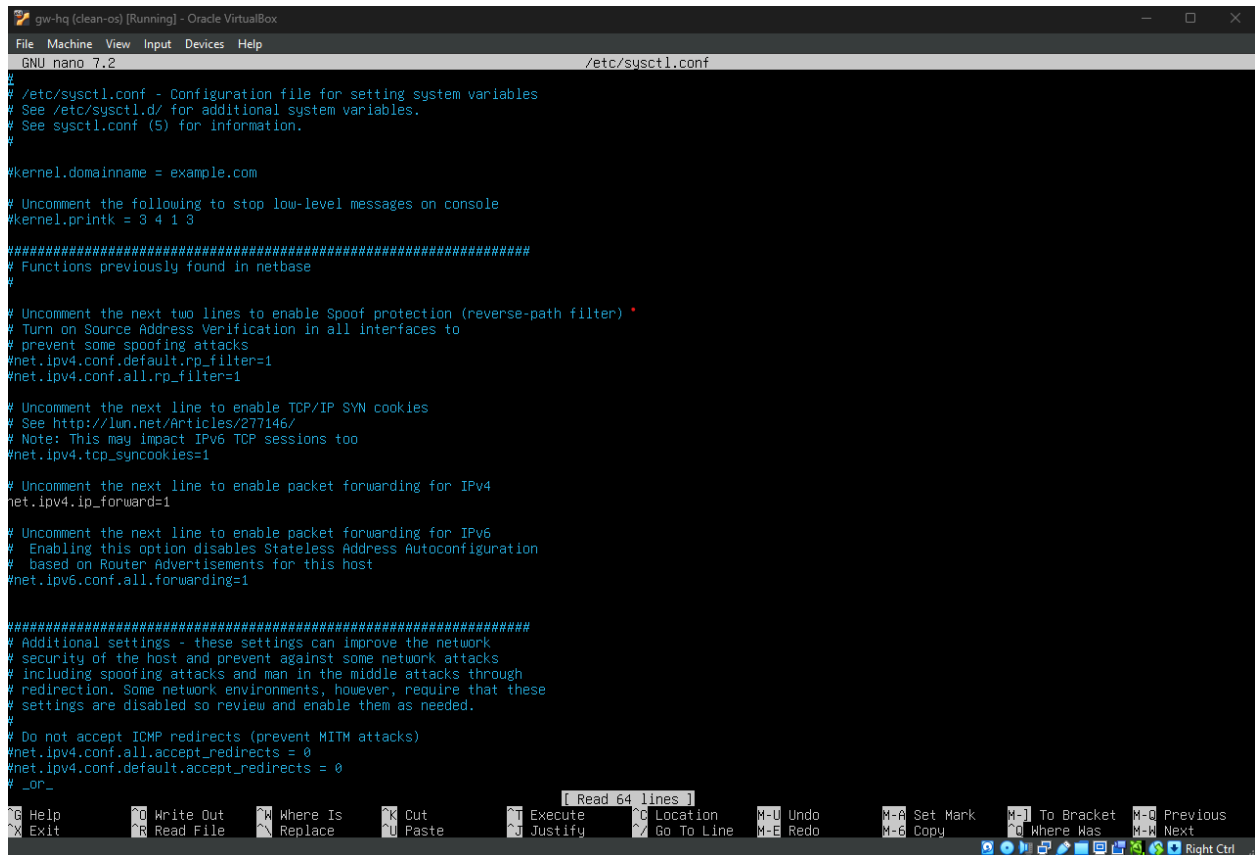
For the gateway machines to correctly route traffic between their respective LANs and the PublicNet through the IPSec tunnel, IP forwarding must be enabled.

```
sudo sysctl -w net.ipv4.ip_forward=1
```

To ensure this setting persists across reboots, the /etc/sysctl.conf file was edited:

```
sudo nano /etc/sysctl.conf
```

The line net.ipv4.ip_forward=1 was uncommented (by removing the # symbol).

A screenshot of a terminal window titled 'gw-hq (clean-os) [Running] - Oracle VirtualBox'. The terminal shows the nano 7.2 editor editing the file /etc/sysctl.conf. The file contains various system configuration settings, including kernel.domainname, kernel.printk, and network-related settings for IPv4 and IPv6. The bottom of the window shows a menu bar with options like Help, Write Out, Where Is, Cut, Execute, Location, M-U Undo, M-A Set Mark, M-I To Bracket, M-O Previous, Exit, Read File, Replace, Paste, Justify, Go To Line, M-E Redo, M-C Copy, M-W Where Was, M-X Next, and Right Ctrl.

3.3 strongSwan Main Configuration (/etc/ipsec.conf)

The core IPSec connection parameters are defined in /etc/ipsec.conf on each gateway. The configuration specifies the local and remote IP addresses, subnets, identifiers, and the cryptographic algorithms to be used.

- **gw-hq Configuration (/etc/ipsec.conf):**

config setup

charondebug="all"

uniqueids=no

conn site-to-site

auto=start # Initiate connection automatically on startup

left=192.168.99.4 # HQ's PublicNet IP

leftsubnet=10.0.0.0/24 # HQ's LAN subnet

right=192.168.99.5 # BR's PublicNet IP

rightsubnet=10.1.0.0/24 # BR's LAN subnet


```

keyexchange=ikev2      # Use IKEv2 for key exchange
authby=secret          # Authenticate using a pre-shared key
ike=aes256gcm16-sha256-modp2048 # IKE Phase 1: AES-256 GCM, SHA256,
Diffie-Hellman Group 2048
esp=aes256gcm16-sha256 # IKE Phase 2 (ESP): AES-256 GCM, SHA256
dpdaction=restart      # Restart connection if peer is dead

```

The screenshot shows a terminal window titled 'gw-hq (clean-os) [Running] - Oracle VM VirtualBox'. The window displays the contents of the file `/etc/ipsec.conf` using the `nano` text editor. The configuration includes a `config` section for setup and debugging, and two `conn` sections: `site-to-site` and `sample-with-ca-cert`. The `site-to-site` connection is configured with specific IP addresses and subnets, and the `sample-with-ca-cert` connection is configured with certificates and a peer name.

```

GNU nano 7.2 /etc/ipsec.conf
# ipsec.conf - strongSwan IPsec configuration file
# basic configuration

config setup
    charondebug="ike 2,knl 2,cfg 2"
    # uniqueids = no,

# Add connections here.

# Sample VPN connections

conn site-to-site
    auto=start
    left=192.168.99.4
    leftsubnet=10.0.0.0/24
    # leftcert=selfCert.der
    # leftsendcert=never
    right=192.168.99.5
    rightsubnet=10.1.0.0/24
    # rightcert=peerCert.der
    #
    auto=start
    ike=aes256-sha1-modp1024
    esp=aes256-sha1
    keyexchange=ikev2
    authby=psk
    dpdaction=restart

conn sample-with-ca-cert
    #
    leftsubnet=10.1.0.0/16
    #
    leftcert=myCert.pem
    #
    right=192.168.0.2
    #
    rightsubnet=10.2.0.0/16
    #
    rightid="c=CH, o=Linux strongSwan CN=peer name"
    #
    auto=start

```

- **gw-br Configuration (/etc/ipsec.conf):**

```

config setup
    charondebug="all"
    uniqueids=no

```

conn HQ-to-BR

```

auto=start          # Initiate connection automatically on startup
left=192.166.99.5    # BR's PublicNet IP
leftsubnet=10.1.0.0/24 # BR's LAN subnet

```

```

right=192.168.99.4      # HQ's PublicNet IP
rightsubnet=10.0.0.0/24 # HQ's LAN subnet

keyexchange=ikev2      # Use IKEv2 for key exchange
authby=secret          # Authenticate using a pre-shared key
ike=aes256gcm16-sha256-modp2048 # IKE Phase 1: AES-256 GCM, SHA256,
Diffie-Hellman Group 2048
esp=aes256gcm16-sha256 # IKE Phase 2 (ESP): AES-256 GCM, SHA256
dpdaction=restart      # Restart connection if peer is dead

```

```

gw-br (clean-os) [Running] - Oracle VirtualBox
File Machine View Input Devices Help
# ipsec.conf - strongswan IPsec configuration file
# basic configuration

config setup
    charondebug="ike 2,knl 2, cfg 2"
    # uniqueids = no

# Add connections here.

# Sample VPN connections

conn site-to-site
    auto=start
    left=192.168.99.5
    leftsubnet=10.1.0.0/24
    right=192.168.99.4
    rightsubnet=10.0.0.0/24
    ike=aes256-sha1-modp1024
    esp=aes256-sha1
    keyexchange=ikev2
    authby=psk
    dpdaction=restart

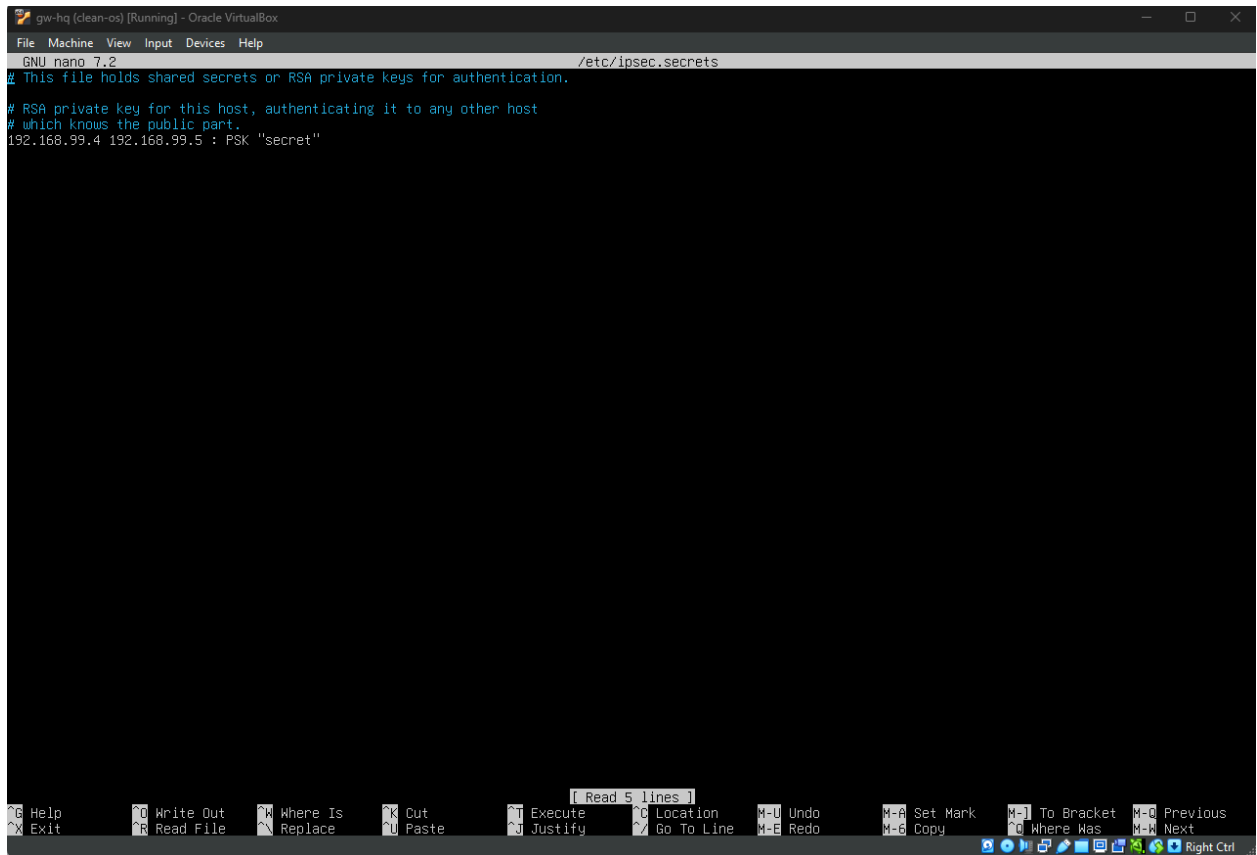
#conn sample-with-ca-cert
#    leftsubnet=10.1.0.0/16
#    leftcert=myCert.pem
#    right=192.168.0.2
#    rightsubnet=10.2.0.0/16
#    rightid="C=CH, O=Linux strongSwan CN=peer name"
#    auto=start

```

3.4 Pre-Shared Key Configuration (/etc/ipsec.secrets)

The pre-shared key (PSK) used for authentication between the two gateways is stored in /etc/ipsec.secrets. This file must contain the exact same key on both gw-hq and gw-br.

- **Configuration (Both Gateways):**
192.168.99.4 192.168.99.5 : PSK "secret"



The screenshot shows a terminal window titled "gw-hq (clean-os) [Running] - Oracle VM VirtualBox". The terminal is running the GNU nano 7.2 text editor, editing the file /etc/ipsec.secrets. The content of the file is as follows:

```
# This file holds shared secrets or RSA private keys for authentication.
#
# RSA private key for this host, authenticating it to any other host
# which knows the public part.
192.168.99.4 192.168.99.5 : PSK "secret"
```

The bottom of the terminal shows the nano editor's command palette with various options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, Copy, To Bracket, Where Was, Previous, Next, and Right Ctrl.

3.5 strongSwan Service Management

After configuring the IPsec files, the strongSwan service was restarted to load the new configurations and enabled to start automatically on system boot.

```
sudo systemctl restart strongswan
sudo systemctl enable strongswan
```

4. IPsec Tunnel Verification

Once the configurations were applied and services restarted, the functionality of the IPsec tunnel was rigorously verified.

4.1 Checking IPsec Status

The `ipsec statusall` command provides a detailed overview of the strongSwan daemon's state and the active Security Associations (SAs).

```
sudo ipsec statusall
```

Expected Output for a Successful Tunnel:

The output should clearly indicate that both the IKE (Phase 1) and ESP (Phase 2) Security Associations are "ESTABLISHED" and "UP". This confirms that the control channel and the data tunnel are successfully negotiated and active.

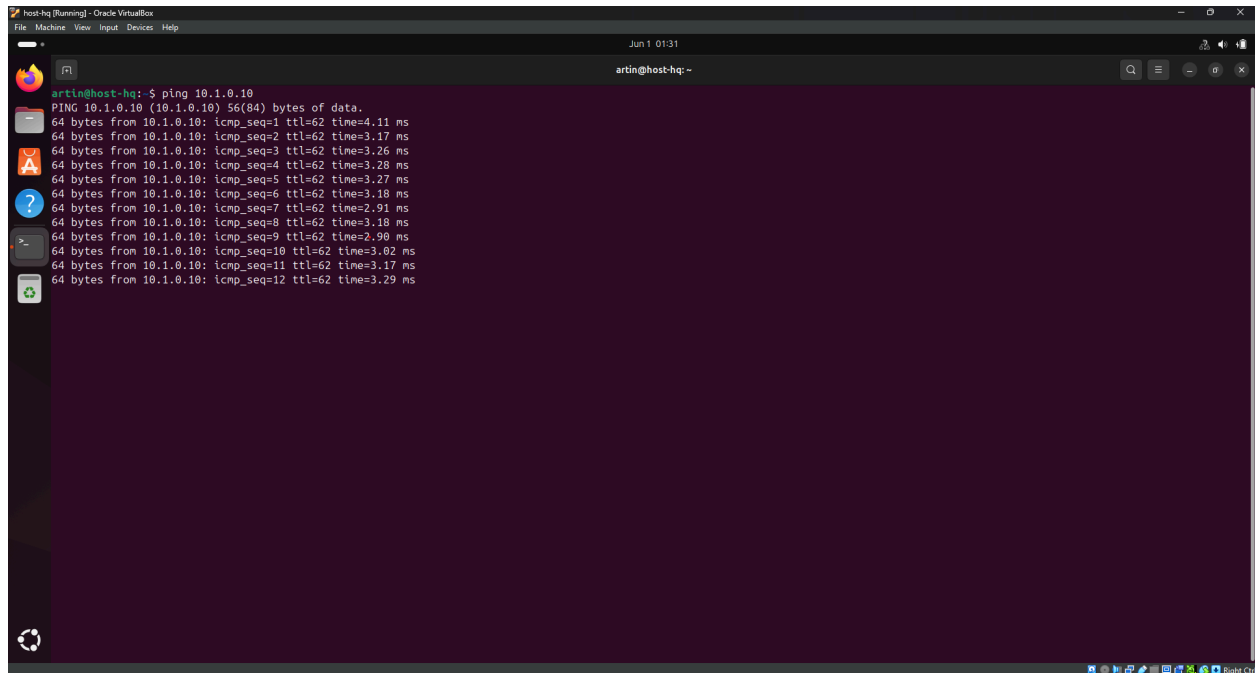
```
root@gw-hq:/home/art# sudo ipsec statusall
Status of IKE charon daemon (strongSwan 5.9.13, Linux 6.8.0-60-generic, x86_64):
  uptime: 68 minutes, since Jun 01 08:03:42 2025
  malloc: sbrk 1880064, mmap 0, used 1071840, free 808224
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 3
  loaded plugins: charon aesni aes rc2 sha2 sha1 md5 mgf1 random nonce x509 revocation constraints pubkey pkcs1 pkcs7 pkcs12 pgp dnskey sshkey pem openssl pkcs8
  fips-prf gmp agent xcbc hmac kdf gcm drbg attr kernel-netlink resolve socket-default connmark stroke updown eap-mschapv2 xauth-generic counters
Listening IP addresses:
  192.168.99.4
  10.0.0.1
Connections:
site-to-site: 192.168.99.4...192.168.99.5 IKEv2, dpddelay=30s
site-to-site: local: [192.168.99.4] uses pre-shared key authentication
site-to-site: remote: [192.168.99.5] uses pre-shared key authentication
site-to-site: child: 10.0.0.0/24 === 10.1.0.0/24 TUNNEL, dpdaction=start
Security Associations (1 up, 0 connecting):
site-to-site{1}: ESTABLISHED 68 minutes ago, 192.168.99.4[192.168.99.4]...192.168.99.5[192.168.99.5]
site-to-site{1}: IKEv2 SPIs: ea4e597041214607_i* da750461be3298aa_r, pre-shared key reauthentication in 95 minutes
site-to-site{1}: IKE proposal: AES_CBC_256/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1024
site-to-site{2}: INSTALLED, TUNNEL, reqid 1, ESP SPIs: c2ac6418_i cbe673f6_o
site-to-site{2}: AES_CBC_256/HMAC_SHA1_96, 0 bytes_i, 0 bytes_o, rekeying in 22 minutes
site-to-site{2}: 10.0.0.0/24 === 10.1.0.0/24
root@gw-hq:/home/art#
```

4.2 End-to-End Connectivity Test (Ping)

The ultimate test for a site-to-site VPN is successful communication between hosts on the separate LANs. This was verified using the ping command.

- From host-hq (10.0.0.10) to host-br (10.1.0.10):
ping 10.1.0.10
- From host-br (10.1.0.10) to host-hq (10.0.0.10):
ping 10.0.0.10

Successful pings confirm that traffic is traversing the IPsec tunnel securely between the two sites.



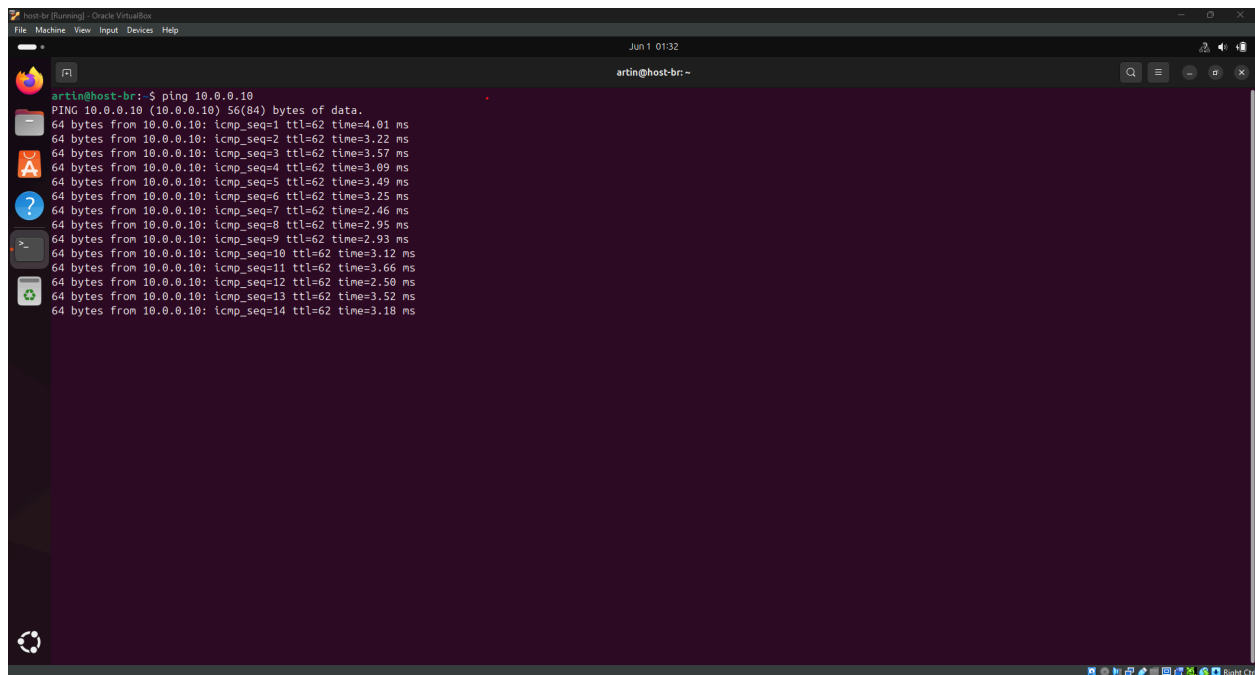
host-hq [Running] - Oracle VM VirtualBox

Jun 1 01:31

artin@host-hq:~\$ ping 10.1.0.10

PING 10.1.0.10 (10.1.0.10) 56(84) bytes of data:

```
64 bytes from 10.1.0.10: icmp_seq=1 ttl=62 time=4.11 ms
64 bytes from 10.1.0.10: icmp_seq=2 ttl=62 time=3.17 ms
64 bytes from 10.1.0.10: icmp_seq=3 ttl=62 time=3.26 ms
64 bytes from 10.1.0.10: icmp_seq=4 ttl=62 time=3.28 ms
64 bytes from 10.1.0.10: icmp_seq=5 ttl=62 time=3.27 ms
64 bytes from 10.1.0.10: icmp_seq=6 ttl=62 time=3.18 ms
64 bytes from 10.1.0.10: icmp_seq=7 ttl=62 time=2.91 ms
64 bytes from 10.1.0.10: icmp_seq=8 ttl=62 time=3.18 ms
64 bytes from 10.1.0.10: icmp_seq=9 ttl=62 time=2.90 ms
64 bytes from 10.1.0.10: icmp_seq=10 ttl=62 time=3.02 ms
64 bytes from 10.1.0.10: icmp_seq=11 ttl=62 time=3.17 ms
64 bytes from 10.1.0.10: icmp_seq=12 ttl=62 time=3.29 ms
```



host-br [Running] - Oracle VM VirtualBox

Jun 1 01:32

artin@host-br:~\$ ping 10.0.0.10

PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data:

```
64 bytes from 10.0.0.10: icmp_seq=1 ttl=62 time=4.01 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=62 time=3.22 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=62 time=3.57 ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=62 time=3.09 ms
64 bytes from 10.0.0.10: icmp_seq=5 ttl=62 time=3.49 ms
64 bytes from 10.0.0.10: icmp_seq=6 ttl=62 time=3.25 ms
64 bytes from 10.0.0.10: icmp_seq=7 ttl=62 time=2.46 ms
64 bytes from 10.0.0.10: icmp_seq=8 ttl=62 time=2.95 ms
64 bytes from 10.0.0.10: icmp_seq=9 ttl=62 time=2.93 ms
64 bytes from 10.0.0.10: icmp_seq=10 ttl=62 time=3.12 ms
64 bytes from 10.0.0.10: icmp_seq=11 ttl=62 time=3.66 ms
64 bytes from 10.0.0.10: icmp_seq=12 ttl=62 time=2.50 ms
64 bytes from 10.0.0.10: icmp_seq=13 ttl=62 time=3.52 ms
64 bytes from 10.0.0.10: icmp_seq=14 ttl=62 time=3.18 ms
```

5. Security Analysis of the IPSec Tunnel

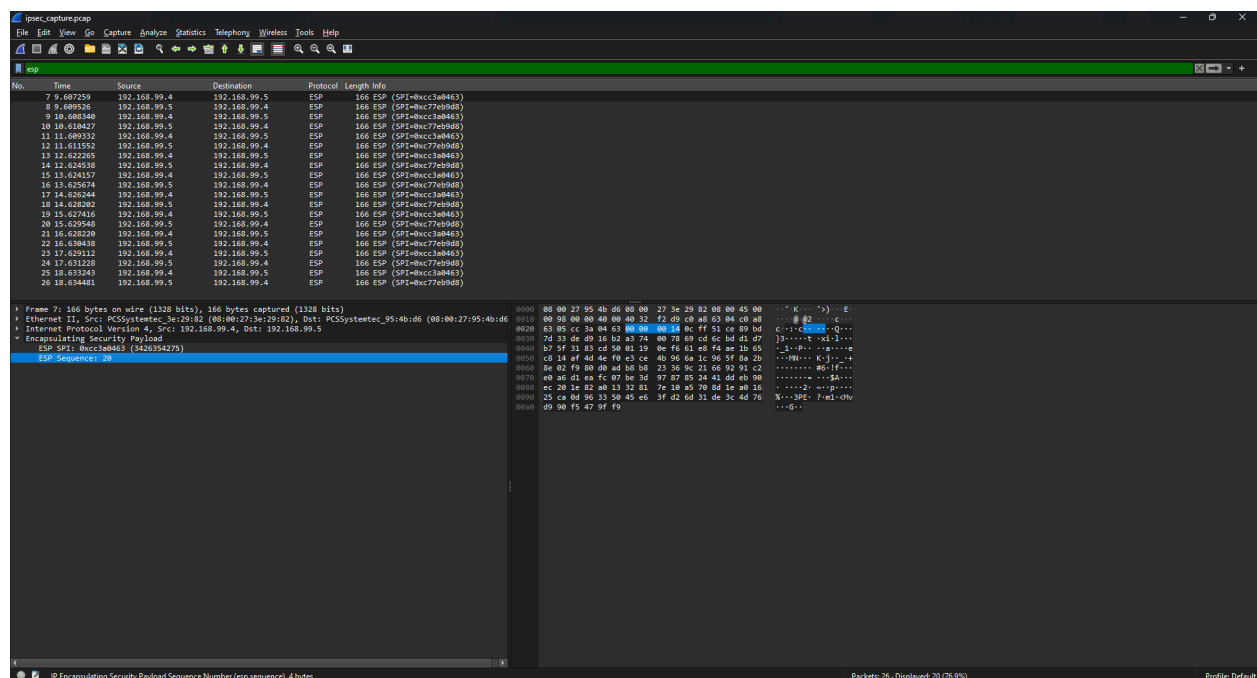
The implemented IPSec tunnel provides robust security features. This section details how to test and analyze its resilience against common network attacks, focusing on the core security properties provided by IPSec.

5.1 Replay Attacks

IPSec incorporates sequence numbers and anti-replay windows within the ESP (Encapsulating Security Payload) protocol. Each encrypted packet is assigned a unique sequence number. If an attacker intercepts and re-sends a packet, the receiving IPSec endpoint will detect the duplicate sequence number (or a number outside the valid window) and discard the packet, thus preventing replay attacks. This mechanism ensures the integrity and freshness of the data.

How to Test for Replay Attack Prevention:

1. **Capture Traffic:** On one of the gateway VMs (e.g., gw-hq), start a tcpdump capture on the PublicNet-facing interface (e.g., ens3):
`sudo tcpdump -i ens3 -w ipsec_capture.pcap host 192.168.99.5 and (udp port 500 or udp port 4500 or proto esp)`
2. **Generate Traffic:** From host-hq, send some continuous traffic to host-br (e.g., `ping 10.1.0.10 -c 10`).
3. **Stop Capture:** Stop the tcpdump capture (Ctrl+C).
4. **Transfer and Analyze:** Transfer ipsec_capture.pcap to a machine with Wireshark (e.g., your host PC or a Kali Linux VM). Open the file in Wireshark.
5. **Identify ESP Packets:** Filter for ESP packets (e.g., `esp or ip.proto == 50`). Observe the sequence numbers in the ESP header.
6. **Attempt Replay (Simulated):** While directly replaying encrypted IPSec packets to bypass the anti-replay mechanism is difficult without the session keys, you can conceptually demonstrate the defense. Using a tool like tcpreplay (often found on Kali Linux or can be installed), you could attempt to re-send a captured ESP packet.
 - **On Kali-attacker VM (connected to PublicNet):**
`sudo apt install tcpreplay`
`sudo tcpreplay --intf1=eth0 ipsec_capture.pcap # Replace eth0 with Kali's PublicNet interface`
 - **Observation:** The strongSwan logs on the receiving gateway (gw-br) should show messages indicating that replayed packets are being dropped or rejected due to anti-replay checks. You will not see the replayed packets successfully reach the host-br.



5.2 Man-in-the-Middle (MitM) Attacks

IPSec's IKE (Internet Key Exchange) protocol is designed to prevent Man-in-the-Middle attacks during the key exchange process. The use of a pre-shared key (PSK) or digital certificates ensures that only authenticated peers can establish the secure tunnel. If an attacker attempts to intercept and modify the key exchange, they will not possess the correct PSK or valid certificate, leading to authentication failure and preventing the tunnel from being established.

How to Test for MitM Prevention (Conceptual):

Directly performing a successful MitM attack against a correctly configured IPSec tunnel with a strong PSK is extremely difficult, as the attacker would need to know the PSK. The test here is primarily conceptual, focusing on the failure scenarios.

1. **Modify PSK on One Side:** As a demonstration of authentication failure, intentionally change the PSK in `/etc/ipsec.secrets` on *only one* of the gateway VMs (e.g., gw-hq).
2. **Restart strongSwan:** Restart strongSwan on both gateways.
3. **Observe Failure:** Check `sudo ipsec statusall` and `sudo journalctl -xeu strongswan` on both gateways.
 - o **Expected Observation:** The tunnel will fail to establish. The logs will show clear "AUTHENTICATION_FAILED" or "INVALID_ID_INFORMATION" messages, indicating that the IKE negotiation failed because the pre-shared keys did not

match. This demonstrates that an unauthorized third party (without the correct PSK) cannot successfully establish a tunnel.

```
root@gw-hq:/home/art# sudo systemctl restart strongswan-starter
root@gw-hq:/home/art# sudo ipsec statusall
Status of IKE charon daemon (strongswan 5.9.19, Linux 6.8.0-60-generic, x86_64):
  uptime: 8 seconds, since Jun 01 09:19:17 2025
  malloc: sbrk 1959594, mmap 0, used 866892, free 992752
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 0
  loaded plugins: charon aesni aes rc2 sha2 sha1 md5 mgf1 random nonce x509 revocation constraints pubkey pkcs1 pkcs7 pkcs12 pgp dnskey sshkey pem openssl pkcs8
  fips-prf gmp agent xcbc hmac kdf gcm drbg attr kernel-netlink resolve socket-default connmark stroke updown eap-mschapv2 xauth-generic counters
Listening IP addresses:
  192.168.99.4
  10.0.0.1
Connections:
site-to-site: 192.168.99.4...192.168.99.5 IKEv2, dpddelay=30s
site-to-site: local: [192.168.99.4] uses pre-shared key authentication
site-to-site: remote: [192.168.99.5] uses pre-shared key authentication
site-to-site: child: 10.0.0/24 == 10.1.0/24 TUNNEL, dpdaction=start
Security Associations (0 up, 0 connecting):
  none
root@gw-hq:/home/art#
```

```
gw-hq (clean-os) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Jun 01 09:19:17 gw-hq charon[3614]: 07[IKE] activating IKE_NATD task
Jun 01 09:19:17 gw-hq charon[3614]: 07[IKE] activating IKE_CERT_PRE task
Jun 01 09:19:17 gw-hq charon[3614]: 07[IKE] activating IKE_AUTH task
Jun 01 09:19:17 gw-hq charon[3614]: 07[IKE] activating IKE_CERT_POST task
Jun 01 09:19:17 gw-hq charon[3614]: 07[IKE] activating IKE_ME task
Jun 01 09:19:17 gw-hq charon[3614]: 07[IKE] activating IKE_CONFIG task
Jun 01 09:19:17 gw-hq charon[3614]: 07[IKE] activating IKE_AUTH_LIFETIME task
Jun 01 09:19:17 gw-hq charon[3614]: 07[IKE] activating IKE_MOBIKE task
Jun 01 09:19:17 gw-hq charon[3614]: 07[IKE] activating IKE_ESTABLISH task
Jun 01 09:19:17 gw-hq charon[3614]: 07[IKE] activating CHILD_CREATE task
Jun 01 09:19:17 gw-hq charon[3614]: 07[IKE] initiating IKE_SA site-to-site[1] to 192.168.99.5
Jun 01 09:19:17 gw-hq charon[3614]: 07[IKE] initiating IKE_SA site-to-site[1] to 192.168.99.5
Jun 01 09:19:17 gw-hq charon[3614]: 07[IKE] IKE_SA site-to-site[1] state change: CREATED => CONNECTING
Jun 01 09:19:17 gw-hq charon[3614]: 12[CFG] configured proposals: IKE:AES_CBC_256/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1024, IKE:AES_CBC_128/AES_CBC_192/AES_CBC_256/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1024, IKE:AES_CBC_128/AES_CBC_192/AES_CBC_256/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_768, IKE:AES_CBC_128/AES_CBC_192/AES_CBC_256/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_512
Jun 01 09:19:17 gw-hq charon[3614]: 07[ENC] generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) N(FRAG_SUP) N(HASH_ALG) N(REDIR_SUP) ]
Jun 01 09:19:17 gw-hq charon[3614]: 07[NET] sending packet: from 192.168.99.4[500] to 192.168.99.5[500] (1080 bytes)
Jun 01 09:19:17 gw-hq charon[3614]: 12[NET] received packet: from 192.168.99.5[500] to 192.168.99.4[500] (344 bytes)
Jun 01 09:19:17 gw-hq charon[3614]: 12[ENC] parsed IKE_SA_INIT response 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) N(FRAG_SUP) N(HASH_ALG) N(CHDLESS_SUP) N(MULT_AUTH) ]
Jun 01 09:19:17 gw-hq charon[3614]: 12[IKE] received FRAGMENTATION_SUPPORTED notify
Jun 01 09:19:17 gw-hq charon[3614]: 12[IKE] received SIGNATURE_HASH_ALGORITHMS notify
Jun 01 09:19:17 gw-hq charon[3614]: 12[IKE] received CHILDLESS_IKEV2_SUPPORTED notify
Jun 01 09:19:17 gw-hq charon[3614]: 12[CFG] selecting proposal:
Jun 01 09:19:17 gw-hq charon[3614]: 12[CFG] proposal matches
Jun 01 09:19:17 gw-hq charon[3614]: 12[CFG] received proposals: IKE:AES_CBC_256/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1024
Jun 01 09:19:17 gw-hq charon[3614]: 12[CFG] configured proposals: IKE:AES_CBC_256/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1024, IKE:AES_CBC_128/AES_CBC_192/AES_CBC_256/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1024, IKE:AES_CBC_128/AES_CBC_192/AES_CBC_256/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_768, IKE:AES_CBC_128/AES_CBC_192/AES_CBC_256/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_512
Jun 01 09:19:17 gw-hq charon[3614]: 12[CFG] selected proposal: IKE:AES_CBC_256/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1024
Jun 01 09:19:17 gw-hq charon[3614]: 12[CFG] received supported signature hash algorithms: sha256 sha384 sha512 identity
Jun 01 09:19:17 gw-hq charon[3614]: 12[IKE] reinitiating already active tasks
Jun 01 09:19:17 gw-hq charon[3614]: 12[IKE] IKE_CERT_PRE task
Jun 01 09:19:17 gw-hq charon[3614]: 12[IKE] IKE_AUTH task
Jun 01 09:19:17 gw-hq charon[3614]: 12[IKE] authentication of '192.168.99.4' (myself) with pre-shared key
Jun 01 09:19:17 gw-hq charon[3614]: 12[IKE] successfully created shared key MAC
Jun 01 09:19:17 gw-hq charon[3614]: 12[CFG] proposing traffic selectors for us:
Jun 01 09:19:17 gw-hq charon[3614]: 12[CFG] 10.0.0/24
Jun 01 09:19:17 gw-hq charon[3614]: 12[CFG] proposing traffic selectors for other:
Jun 01 09:19:17 gw-hq charon[3614]: 12[CFG] 10.1.0/24
Jun 01 09:19:17 gw-hq charon[3614]: 12[CFG] configured proposals: ESP:AES_CBC_256/HMAC_SHA1_96/NO_EXT_SEQ, ESP:AES_CBC_128/AES_CBC_192/AES_CBC_256/HMAC_SHA1_96/NO_EXT_SEQ, ESP:AES_CBC_128/AES_CBC_192/AES_CBC_256/HMAC_SHA1_96/NO_EXT_SEQ, ESP:AES_CBC_128/AES_CBC_192/AES_CBC_256/HMAC_SHA1_96/NO_EXT_SEQ
Jun 01 09:19:17 gw-hq charon[3614]: 12[IKE] establishing CHILD_SA site-to-site[1]
Jun 01 09:19:17 gw-hq charon[3614]: 12[IKE] establishing CHILD_SA site-to-site[1]
Jun 01 09:19:17 gw-hq charon[3614]: 12[KNL] got SPI cf3f6f8d
Jun 01 09:19:17 gw-hq charon[3614]: 12[ENC] generating IKE_AUTH request 1 [ IDi N(INIT_CONTACT) IDr AUTH SA TSi TSr N(MOBIKE_SUP) N(ADD_4_ADDR) N(MULT_AUTH) N(REDIR_SUP) ]
Jun 01 09:19:17 gw-hq charon[3614]: 12[NET] sending packet: from 192.168.99.4[4500] to 192.168.99.5[4500] (412 bytes)
Jun 01 09:19:17 gw-hq charon[3614]: 13[NET] received packet: from 192.168.99.5[4500] to 192.168.99.4[4500] (76 bytes)
Jun 01 09:19:17 gw-hq charon[3614]: 13[ENC] parsed IKE_AUTH response 1 [ N(AUTH_FAILED) ]
Jun 01 09:19:17 gw-hq charon[3614]: 13[IKE] received AUTHENTICATION_FAILED notify error
Jun 01 09:19:17 gw-hq charon[3614]: 13[KNL] deleting SAD entry with SPI cf3f6f8d
Jun 01 09:19:17 gw-hq charon[3614]: 13[KNL] deleted SAD entry with SPI cf3f6f8d
Jun 01 09:19:17 gw-hq charon[3614]: 13[IKE] IKE_SA site-to-site[1] state change: CONNECTING => DESTROYING
lines 978-1026/1026 (END)
```

5.3 Data Confidentiality and Integrity

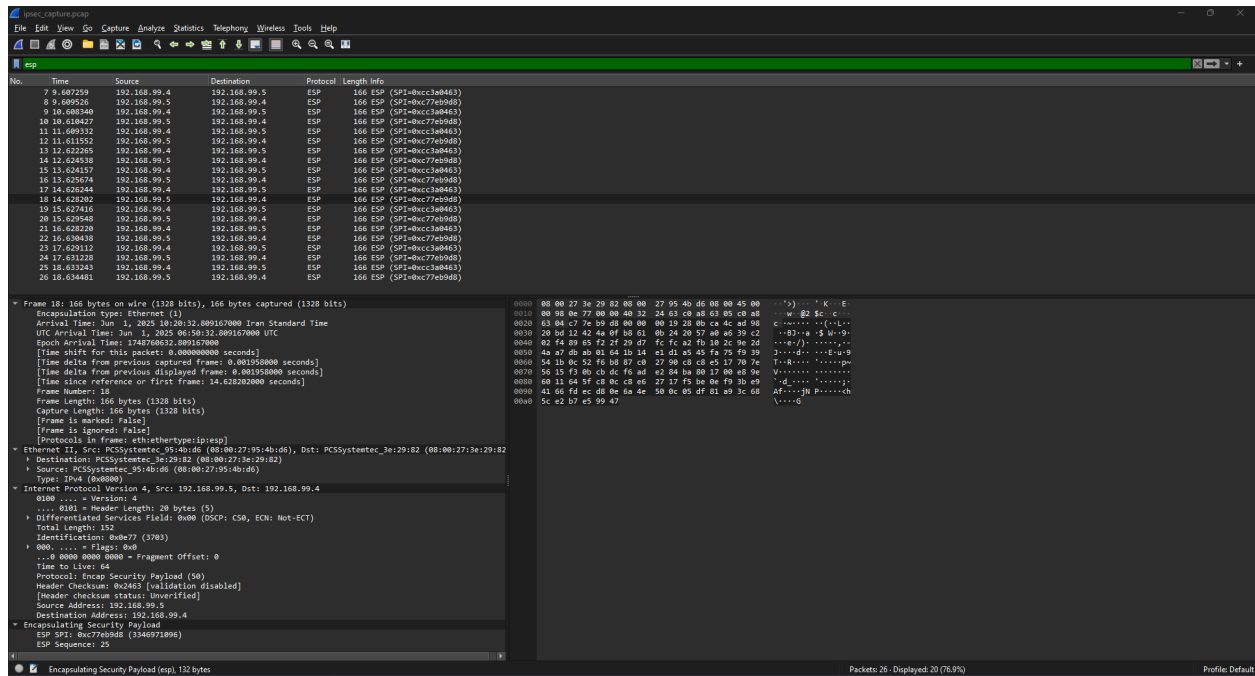
The chosen cryptographic algorithms provide strong protection for data:

- **Confidentiality (Encryption):** AES256-GCM (Advanced Encryption Standard with Galois/Counter Mode, 256-bit key) is a highly secure, authenticated encryption algorithm. It encrypts the data payload, making it unreadable to unauthorized parties.

- **Integrity (Authentication):** SHA256 (Secure Hash Algorithm 256-bit) provides data integrity. It generates a unique hash of the packet, which is verified by the receiver. Any tampering with the data during transit will result in a mismatch of the hash, indicating that the data's integrity has been compromised. GCM mode also inherently provides authentication alongside encryption.

How to Test for Confidentiality and Integrity:

1. **Capture Encrypted Traffic:** On one of the gateway VMs (e.g., gw-hq), start a tcpdump capture on the PublicNet-facing interface (ens3) while traffic is flowing through the tunnel:
`sudo tcpdump -i ens3 -w encrypted_traffic.pcap host 192.168.99.5 and proto esp`
2. **Generate Traffic:** From host-hq, send some data to host-br (e.g., ping 10.1.0.10 -c 10 or nc -zvw 10.1.0.10 80 if you have a web server).
3. **Stop Capture:** Stop the tcpdump capture.
4. **Analyze in Wireshark:** Transfer encrypted_traffic.pcap to a machine with Wireshark.
 - **Confidentiality Verification:** Observe the ESP packets. You will notice that the payload of these packets is encrypted and unreadable. Wireshark will likely show it as "Encrypted Data" or similar, without being able to dissect the inner IP or ICMP headers (unless you configure Wireshark with the IPSec keys, which is usually not done for this type of demonstration). This confirms confidentiality.
 - **Integrity Verification:** While you can't directly "break" integrity without the key, the presence of the ESP header and the successful decryption by strongSwan implies that the integrity check (part of GCM) passed. If an attacker were to tamper with the packet, strongSwan would detect the integrity failure and drop the packet, logging an error.



6. Performance Analysis

While security is paramount, the overhead introduced by encryption and decryption can impact network performance. This section outlines methods for analyzing the tunnel's performance.

6.1 Throughput Measurement

iperf3 is a widely used tool for measuring network bandwidth and throughput. It can be used to compare the network speed with and without the IPsec tunnel.

- **Setup:**
 - Install iperf3 on host-hq and host-br: `sudo apt install iperf3`
 - On host-br (server): `iperf3 -s`
 - On host-hq (client): `iperf3 -c 10.1.0.10`
- **Analysis:** Compare the bandwidth achieved when the IPsec tunnel is active versus when it is inactive. The IPsec tunnel will typically show a reduction in throughput due to the cryptographic operations.

6.2 Latency Measurement

The ping command can also be used to measure latency (round-trip time) between the hosts.

- **Analysis:** Compare the average ping times between host-hq and host-br when

the IPSec tunnel is active versus when it is inactive. Encryption/decryption overhead will typically introduce a slight increase in latency.

6.3 CPU Utilization on Gateways

The cryptographic operations performed by IPSec are CPU-intensive. Monitoring CPU utilization on the gateway machines (gw-hq and gw-br) during high traffic loads (e.g., during iperf3 tests) is crucial.

- **Monitoring:** Use tools like top or htop on the gateway VMs to observe CPU usage.
- **Analysis:** Higher CPU utilization on the gateways during tunnel activity indicates the overhead of IPSec processing.

7. Troubleshooting Common Issues

During the implementation, several common issues can arise. Understanding these and their solutions is vital for successful deployment.

7.1 Firewall Configuration

- **Symptom:** IKE (Phase 1) might establish, but ESP (Phase 2) fails, or no connection attempts are visible.
- **Cause:** Firewalls (e.g., ufw or iptables) on the gateway machines blocking IPSec-related traffic.
- **Solution:** Ensure UDP ports 500 (for IKE) and 4500 (for NAT-T encapsulated ESP) are open. Also, ensure the ESP protocol (IP Protocol 50) is allowed.

```
sudo ufw allow 500/udp  
sudo ufw allow 4500/udp  
sudo ufw allow proto esp  
sudo ufw reload  
sudo ufw status numbered # Verify rules
```

7.2 Configuration Mismatches

- **Symptom:** IKE or ESP fails to establish, often with "NO_PROPOSAL_CHOSEN" or "AUTHENTICATION_FAILED" errors in logs.
- **Cause:** Any mismatch between the ipsec.conf or ipsec.secrets files on the two gateways. This includes:
 - Mismatched ike or esp algorithms.
 - Incorrect left/right IP addresses or leftsubnet/rightsubnet.

- Mismatched leftid/rightid identifiers.
- Incorrect or inconsistent pre-shared key in ipsec.secrets (case-sensitive, sensitive to extra spaces).
- **Solution:** Meticulously compare the configuration files on both gateways, ensuring every parameter is identical where required.

7.3 strongSwan Service Not Initiating Connection

- **Symptom:** ipsec statusall shows "O up, O connecting" even after sudo systemctl restart strongswan.
- **Cause:** The auto=add setting in ipsec.conf only loads the configuration but doesn't necessarily initiate the connection.
- **Solution:** Change auto=add to auto=start in ipsec.conf on both gateways. This forces strongSwan to try to establish the tunnel immediately upon startup.

7.4 Routing Issues

- **Symptom:** IPsec tunnel establishes, but hosts on the LANs cannot ping each other.
- **Cause:** Incorrect IP forwarding settings on gateways, or incorrect default gateways on LAN hosts.
- **Solution:**
 - Verify net.ipv4.ip_forward=1 is enabled on gateways.
 - Ensure LAN hosts have their respective gateway's LAN IP configured as their default gateway. Use ip route show on all VMs to verify.

7.5 Debugging Tools

- **sudo ipsec statusall:** Provides a quick overview of the tunnel's state.
- **sudo journalctl -xeu strongswan:** The most important tool for detailed strongSwan logs. Look for error messages, negotiation failures, or authentication issues.
- **sudo tcpdump -n -i <interface> port 500 or port 4500 or proto esp:** Captures raw network traffic related to IPsec. Invaluable for seeing if packets are being sent, received, or dropped.

8. Conclusion

This project successfully demonstrated the design, implementation, and verification of an IPsec site-to-site VPN tunnel using strongSwan in a virtualized environment. The tunnel effectively provides confidentiality, integrity, and authentication for data communications between the simulated Headquarters and Branch Office networks. Through systematic configuration and troubleshooting, a secure and functional network overlay was established. The project also highlighted the importance of

meticulous configuration, robust firewall management, and effective debugging techniques for deploying secure network solutions.

9. Suggestions for Future Work

- **Certificate-Based Authentication:** Implement IPsec using X.509 certificates instead of pre-shared keys for enhanced security and scalability.
- **Advanced Routing:** Integrate dynamic routing protocols (e.g., OSPF, BGP) over the IPsec tunnel.
- **Intrusion Detection/Prevention:** Deploy IDS/IPS tools (e.g., Snort, Suricata) to monitor traffic traversing the tunnel for malicious activity.
- **Performance Optimization:** Explore hardware offloading for cryptographic operations or fine-tune MTU settings for improved throughput.
- **High Availability:** Implement redundant IPsec tunnels for fault tolerance.