Network Security Lab Report:

Student Name: Assem Chebly

Course Name: Network Security

Date: 3 Novembre 2025

Part 1 - Network Segmentation with Firewall Rules

1. **Lab Environment / Setup:**

```
PS C:\Users\assem\Downloads\TP2-code\TP2-firewall> docker compose ps
NAME                                IMAGE                                                      COMMAND                  SERVICE             CREATED          STATUS          PORTS
tp2-firewall-developerA-1           jackops93/netsetalpine:v1.0.0                              "sh -c 'ip route add…"   developerA          15 minutes ago   Up 14 minutes
tp2-firewall-developerB-1           jackops93/netsetalpine:v1.0.0                              "sh -c 'ip route add…"   developerB          15 minutes ago   Up 14 minutes
tp2-firewall-devserver-1            jackops93/netsecnginx:v1.0.0                               "sh -c 'ip route add…"   devserver           15 minutes ago   Up 14 minutes   80/tcp
tp2-firewall-gsiteA-1               jackops93/netsetalpine:v1.0.0                              "sh -c 'sysctl -w ne…"   gsiteA              15 minutes ago   Up 14 minutes
tp2-firewall-gsiteB-1               jackops93/netsetalpine:v1.0.0                              "sh -c 'sysctl -w ne…"   gsiteB              15 minutes ago   Up 14 minutes
tp2-firewall-gwcloudA-1             jackops93/netsetalpine:v1.0.0                              "sh -c 'sysctl -w ne…"   gwcloudA            15 minutes ago   Up 14 minutes
tp2-firewall-gwcloudB-1             jackops93/netsetalpine:v1.0.0                              "sh -c 'sysctl -w ne…"   gwcloudB            15 minutes ago   Up 14 minutes
tp2-firewall-productionserver-1     jackops93/netsecnginx:v1.0.0                               "sh -c 'ip route add…"   productionserver    15 minutes ago   Up 14 minutes   80/tcp
tp2-firewall-router-1               jackops93/netsetalpine:v1.0.0                              "sh -c 'sysctl -w ne…"   router              15 minutes ago   Up 14 minutes
tp2-firewall-secureserver-1         registry.gitlab.com/dasskelett/nginx-quic-docker/nginx-quic:latest   "sh -c '\n  ip route …"   secureserver        15 minutes ago   Up 14 minutes   80/tcp, 443/tcp, 443/udp
tp2-firewall-staffA-1               jackops93/netsetalpine:v1.0.0                              "sh -c 'ip route add…"   staffA              15 minutes ago   Up 14 minutes
tp2-firewall-staffB-1               jackops93/netsetalpine:v1.0.0                              "sh -c 'ip route add…"   staffB              15 minutes ago   Up 14 minutes
```

As shown in Figure 1, each office, cloud site, and resource runs in a separate container managed by Docker Compose, allowing fine-grained control over network topology for firewall testing.

2. **Network Topology:**
   The lab environment consists of:

   Office Site A (gsiteA):
   Internal network 10.10.0.0/16
   Developer A: 10.10.0.10
   Gateway interface: 10.10.0.3

   Office Site B (gsiteB):
   Internal network 10.11.0.0/16
   Developer B: 10.11.0.10
   Staff B: 10.11.0.11
   Gateway interface: 10.11.0.3

   Cloud Site A (gwcloudA):
   Internal network 10.12.0.0/16
   Production Server: 10.12.0.30
   Development Server: 10.12.0.20
   Gateway interface: 10.12.0.3

   Cloud Site B (gwcloudB):
   Internal network 10.13.0.0/16

Secure Server: 10.13.0.40
Gateway interface: 10.13.0.3

3. **Firewall Architecture and Policy**

Access Control Requirements:

Requirement 1: All office users can access Production server

- Source: 10.10.0.0/16 (Office A) and 10.11.0.0/16 (Office B)
- Destination: 10.12.0.30 (Production)
- Action: ACCEPT

```
gsiteA:/# iptables -L -v -n
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 7 packets, 2756 bytes)
 pkts bytes target     prot opt in     out     source               destination
    7   451 ACCEPT     all  --  *      *       10.10.0.0/16         10.12.0.30

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Requirement 2: Developers can access Secure server from any location

- Source: 10.10.0.10 (Developer A) and 10.11.0.10 (Developer B)
- Destination: 10.13.0.40 (Secure Server)
- Action: ACCEPT

Requirement 3: Staff in Office B can access both Production and Secure servers

- Source: 10.11.0.11 (Staff B)
- Destination: 10.12.0.30 (Production) and 10.13.0.40 (Secure)
- Action: ACCEPT

Requirement 4: Developer A can access Development server in Cloud

- Source: 10.10.0.10 (Developer A)
- Destination: 10.12.0.20 (Development)

- Action: ACCEPT

Default Action: All other traffic is DROPPED

4. Firewall Configuration

4.1 Configuration Methodology

Each gateway container was configured using the following procedure:

1. Flush existing rules: iptables -F removes all custom rules while preserving built-in chains.
2. Set default policies: iptables -P FORWARD DROP establishes default-deny posture.
3. Add ACCEPT rules: Rules for legitimate traffic added with highest priority.
4. Export configuration: iptables-save > filename.txt documents.

```
gsiteA:/# iptables -F
gsiteA:/# iptables -P FORWARD DROP
gsiteA:/# iptables -A FORWARD -s 10.10.0.0/16 -d 10.12.0.30 -j ACCEPT
```

4.2 gsiteA (Office Site A Gateway) - Configuration

Configuration Commands:

- Iptables –F
- Iptables –P FORWARD DROP
- iptables -A FORWARD -s 10.10.0.0/16 -d 10.12.0.30 -j ACCEPT
- iptables -A FORWARD -s 10.10.0.10 -d 10.13.0.40 -j ACCEPT
- iptables -A FORWARD -s 10.10.0.10 -d 10.12.0.20 -j ACCEPT

Exported Configuration (assem_gsiteA.txt)

```
gsiteA:/# iptables -L -v -n
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 7 packets, 2756 bytes)
 pkts bytes target     prot opt in     out     source               destination
    7   451 ACCEPT     all  --  *      *       10.10.0.0/16         10.12.0.30
    0     0 ACCEPT     all  --  *      *       10.10.0.10           10.13.0.40
    0     0 ACCEPT     all  --  *      *       10.10.0.10           10.12.0.20

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

4.3 gsiteB (Office Site B Gateway) - Configuration

Configuration Commands:

- iptables -F
- iptables -P FORWARD DROP
- iptables -A FORWARD -s 10.11.0.0/16 -d 10.12.0.30 -j ACCEPT
- iptables -A FORWARD -s 10.11.0.10 -d 10.13.0.40 -j ACCEPT
- iptables -A FORWARD -s 10.11.0.11 -d 10.12.0.30 -j ACCEPT
- iptables -A FORWARD -s 10.11.0.11 -d 10.13.0.40 -j ACCEPT
- iptables-save > assem_gsiteB.txt

```
gsiteB:/# iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 8 packets, 2816 bytes)
 pkts bytes target     prot opt in     out     source               destination
    7   451 ACCEPT     all  --  *      *       10.11.0.0/16         10.12.0.30
    0     0 ACCEPT     all  --  *      *       10.11.0.10           10.13.0.40
    0     0 ACCEPT     all  --  *      *       10.11.0.11           10.12.0.30
    0     0 ACCEPT     all  --  *      *       10.11.0.11           10.13.0.40

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

5.Testing and Verification

Connectivity Tests:

Connectivity verification was performed using custom shell scripts checking reachability from each office site to cloud resources

```
staffB:/# ./scripts/check_reachability.sh
Connection to production server is possible.
Unable to connect to development server.
Connection to secure server is possible.
```

```
developerA:/# ./scripts/c
bash: ./scripts/c: No such file or directory
developerA:/# ./scripts/check_reachability.sh
Connection to production server is possible.
Connection to development server is possible.
Connection to secure server is possible.
developerA:/# exit
exit
PS C:\Users\assem\Downloads\TP2-code\TP2-firewall> docker compose exec -it developerB bash
developerB:/# ./scripts/check_reachability.sh
Connection to production server is possible.
Unable to connect to development server.
Connection to secure server is possible.
```

```
staffA:/# ./scripts/check_reachability.sh
Connection to production server is possible.
Unable to connect to development server.
Unable to connect to secure server.
```

6. Configuration Export and Reproducibility

All firewall configurations have been exported in iptables-save format

Exported Files:

- Assem_gsiteA.txt
- Assem_gsiteB.txt
- Assem_gwcloudA.txt
- Assem_gwcloudB.txt

Restoration Procedure:

- iptables-restore < assem_gsiteA.txt
- iptables-save > /etc/iptables/rules.v4

This allows identical firewall policies to be replicated across multiple deployments or environments.

Part 2 - IPSec Tunnels

1. **IPSec Overview**
   IPSec is a suite of protocols that provide security at the IP layer by authenticating and encrypting each IP packet. It operates in two modes:

- **Tunnel Mode:** Encapsulates the entire original packet, adding a new IP header and security information. This mode is used for site-to-site VPN tunnels.

- **Transport Mode:** Only the payload is encrypted, leaving the original IP header intact. Used for host-to-host communication.

For this lab, tunnel mode is utilized, as it allows network-level security and hides internal topology information from external observers

2. **Tunnel Characteristics**
   - **Protocol**: IKEv2 (Internet Key Exchange version 2) for key negotiation 10.12.0.0/16 cloudA-officeA/B
   - **Encryption**: AES-256-CBC (Advanced Encryption Standard, 256-bit key, Cipher Block Chaining mode)
   - **Integrity:** HMAC-SHA2-256 (Hash-based Message Authentication Code with SHA-256)
   - **Key Exchange Group:** MODP2048 (Diffie-Hellman group 14)
   - **Tunnel Mode:** Tunnel mode (site-to-site), authenticated with pre-shared keys (PSK)
   - **Dead Peer Detection (DPD):** Enabled with 30-second intervals and restart action on peer failure

3. **Configuration Details**

   3.1 StrongSwan Installation
   StrongSwan is an open-source implementation of IPSec, providing comprehensive VPN functionality on Linux systems
   - apt-get update
   - apt-get install strongswan

   3.2 IPSec Configuration File (/etc/ipsec.conf)

   - Configuration Explanation: config setup: Global daemon settings enabling moderate-level debugging for all IPSec components

   - uniqueids=no: Allows multiple simultaneous tunnels with potentially overlapping IDs

- keyexchange=ikev2: Uses IKEv2 protocol for more efficient and secure key negotiation
- left/right: Defines tunnel endpoints (gateway IPs) from the perspective of this gateway
- leftsubnet/rightsubnet: Specifies which internal networks are protected by the tunnel
- leftid/rightid: Peer identifiers used for authentication and logging
- ike/esp: Cryptographic algorithm specifications with enforcement (! flag) dpdaction=restart: Automatically re-establishes tunnel if peer becomes unreachable
- auto=add: Loads connection profile at daemon startup but requires manual activation

3.2.1 The following configuration was deployed on **gsiteA** (Office A Gateway):

```
# ipsec.conf - strongSwan IPsec configuration file

# basic configuration

config setup
        # strictcrlpolicy=yes
        # uniqueids = no

# Add connections here.

conn officeA-cloudA
        keyexchange=ikev2

        left=10.10.0.3
        leftsubnet=10.10.0.0/16
        leftid=@officeA
        leftauth=psk

        right=10.12.0.3
        rightsubnet=10.12.0.0/16
        rightid=@cloudA
        rightauth=psk

        ike=aes256-sha256-modp2048!
        esp=aes256-sha256!

        auto=add

#Sample VPN connections
```

3.2.2 The following configuration was deployed on **gsiteB** (Office B Gateway):

```
# ipsec.conf - strongSwan IPsec configuration file

# basic configuration

config setup
        # strictcrlpolicy=yes
        # uniqueids = no

# Add connections here.

conn officeB-cloudA
        keyexchange=ikev2

        left=10.11.0.3
        leftsubnet=10.11.0.0/16
        leftid=@officeB
        leftauth=psk

        right=10.12.0.3
        rightsubnet=10.12.0.0/16
        rightid=@cloudA
        rightauth=psk

        ike=aes256-sha256-modp2048!
        esp=aes256-sha256!

        auto=add

# Sample VPN connections
```

3.2.3 The following configuration was deployed on gwcloudA(cloud A Gateway)

```
# ipsec.conf - strongSwan IPsec configuratio

# basic configuration

config setup
        # strictcrlpolicy=yes
        # uniqueids = no

# Add connections here.

conn cloudA-officeA
        keyexchange=ikev2

        left=10.12.0.3
        leftsubnet=10.12.0.0/16
        leftid=@cloudA
        leftauth=psk

        right=10.10.0.3
        rightsubnet=10.10.0.0/16
        rightid=@officeA
        rightauth=psk

        ike=aes256-sha256-modp2048!
        esp=aes256-sha256!
        auto=add

conn cloudA-officeB
        keyexchange=ikev2

        left=10.12.0.3
        leftsubnet=10.12.0.0/16
        leftid=@cloudA
        leftauth=psk

        right=10.11.0.3
        rightsubnet=10.11.0.0/16
        rightid=@officeB
        rightauth=psk

        ike=aes256-sha256-modp2048!
        esp=aes256-sha256!
        auto=add

# Sample VPN connections
```

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

Both gateway pairs share a common PSK for mutual authentication:

GsiteA:

```
# ipsec.secrets - strongSwan IPsec secrets file
@officeA @cloudA : PSK "NETSEC_TP2"
```

GsiteB:

```
# ipsec.secrets - strongSwan IPsec secrets file
@officeB @cloudA : PSK "NETSEC_TP2_SECRETKEY_OFFICEB_CLOUDA"
```

GwcloudA:

```
# ipsec.secrets - strongSwan IPsec secrets file
@cloudA @officeA : PSK "NETSEC_TP2"
@cloudA @officeB : PSK "NETSEC_TP2_SECRETKEY_OFFICEB_CLOUDA"
```

The PSK is used only for initial authentication during IKE Phase 1 negotiation. Subsequent data encryption uses independently negotiated session keys derived from the IKE exchange.

4. Tunnel Establishment Process

4.1 Step-by-Step Tunnel Activation

Step 1: Restart the Service

Command on gsiteA, gsiteB and gwcloudA:

```
gsiteA:/# ipsec restart
Stopping strongSwan IPsec...
Starting strongSwan 5.9.14 IPsec [starter]...
gsiteA:/#
```

```
gsiteB:/# ipsec restart
Stopping strongSwan IPsec...
Starting strongSwan 5.9.14 IPsec [starter]...
gsiteB:/#
```

```
gwcloudA:/# ipsec restart
Stopping strongSwan IPsec...
Starting strongSwan 5.9.14 IPsec [starter]...
gwcloudA:/#
```

This command reloads configuration files and restarts the charon daemon, loading all defined connection profiles.

Step 2: Initiate Tunnel from officeA to cloudA and from officeB to cloudA:

```
gsiteA:/# ipsec up officeA-cloudA
initiating IKE_SA officeA-cloudA[1] to 10.12.0.3
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) ]
sending packet: from 10.10.0.3[500] to 10.12.0.3[500] (464 bytes)
received packet: from 10.12.0.3[500] to 10.10.0.3[500] (472 bytes)
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_S
UP) N(MULT_AUTH) ]
selected proposal: IKE:AES_CBC_256/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/MODP_2048
authentication of 'officeA' (myself) with pre-shared key
establishing CHILD_SA officeA-cloudA{1}
generating IKE_AUTH request 1 [ IDi N(INIT_CONTACT) IDr AUTH SA TSi TSr N(MOBIKE_SUP) N(NO_ADD_ADDR) N
(MULT_AUTH) N(EAP_ONLY) N(MSG_ID_SYN_SUP) ]
sending packet: from 10.10.0.3[4500] to 10.12.0.3[4500] (288 bytes)
received packet: from 10.12.0.3[4500] to 10.10.0.3[4500] (240 bytes)
parsed IKE_AUTH response 1 [ IDr AUTH SA TSi TSr N(MOBIKE_SUP) N(NO_ADD_ADDR) ]
authentication of 'cloudA' with pre-shared key successful
peer supports MOBIKE
IKE_SA officeA-cloudA[1] established between 10.10.0.3[officeA]...10.12.0.3[cloudA]
scheduling reauthentication in 9797s
maximum IKE_SA lifetime 10337s
selected proposal: ESP:AES_CBC_256/HMAC_SHA2_256_128/NO_EXT_SEQ
CHILD_SA officeA-cloudA{1} established with SPIs c4d5ec0d_i c2754043_o and TS 10.10.0.0/16 === 10.12.0
.0/16
connection 'officeA-cloudA' established successfully
gsiteA:/#
```

```
gsiteB:/# ipsec up officeB-cloudA
initiating IKE_SA officeB-cloudA[1] to 10.12.0.3
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) ]
sending packet: from 10.11.0.3[500] to 10.12.0.3[500] (464 bytes)
received packet: from 10.12.0.3[500] to 10.11.0.3[500] (472 bytes)
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_S
UP) N(MULT_AUTH) ]
selected proposal: IKE:AES_CBC_256/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/MODP_2048
authentication of 'officeB' (myself) with pre-shared key
establishing CHILD_SA officeB-cloudA{1}
generating IKE_AUTH request 1 [ IDi N(INIT_CONTACT) IDr AUTH SA TSi TSr N(MOBIKE_SUP) N(NO_ADD_ADDR) N
(MULT_AUTH) N(EAP_ONLY) N(MSG_ID_SYN_SUP) ]
sending packet: from 10.11.0.3[4500] to 10.12.0.3[4500] (288 bytes)
received packet: from 10.12.0.3[4500] to 10.11.0.3[4500] (240 bytes)
parsed IKE_AUTH response 1 [ IDr AUTH SA TSi TSr N(MOBIKE_SUP) N(NO_ADD_ADDR) ]
authentication of 'cloudA' with pre-shared key successful
peer supports MOBIKE
IKE_SA officeB-cloudA[1] established between 10.11.0.3[officeB]...10.12.0.3[cloudA]
scheduling reauthentication in 10137s
maximum IKE_SA lifetime 10677s
selected proposal: ESP:AES_CBC_256/HMAC_SHA2_256_128/NO_EXT_SEQ
CHILD_SA officeB-cloudA{1} established with SPIs c3903c13_i cbe0cfca_o and TS 10.11.0.0/16 === 10.12.0
.0/16
connection 'officeB-cloudA' established successfully
gsiteB:/#
```

Process Analysis:

1. IKE Phase 1 (IKE_SA_INIT): Gateways negotiate encryption parameters, exchange public key material (Diffie Hellman), and establish a secure channel for further communication
2. Peer Authentication: Both gateways authenticate using their PSK
3. IKE Phase 2 (IKE_AUTH): Child SA for actual data traffic is negotiated
4. CHILD_SA Establishment: Symmetric session keys (SPIs) are created for site-to-site ESP encryption
5. Completion: Traffic between 10.10.0.0/16 and 10.12.0.0/16 and between 10.11.0.0/16 and 10.12.0.0/16 is now encrypted through the tunnels.

4.2 Tunnel Status Verification

Command on gwcloudA:

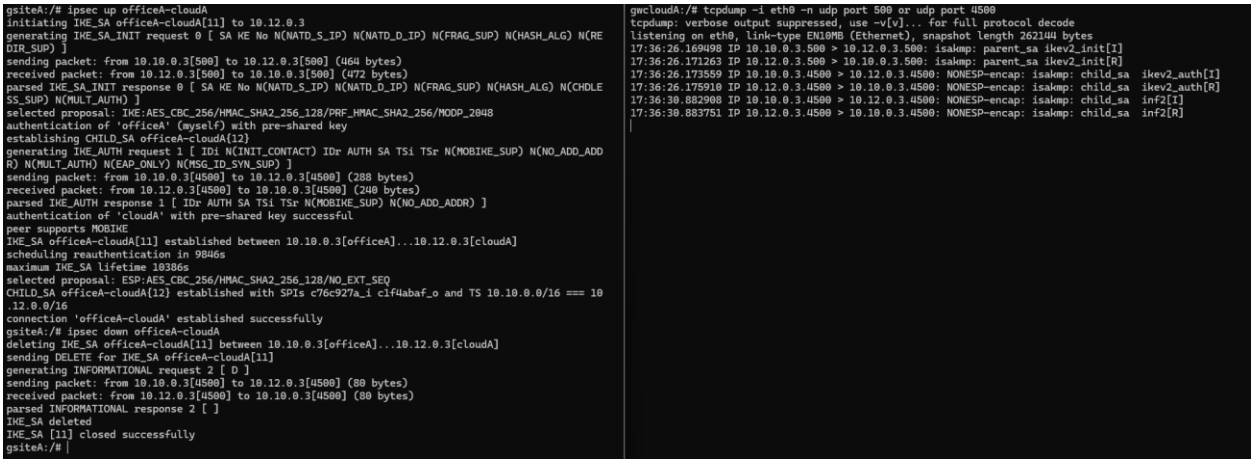
- ipsec statusall

Status Interpretation:

- IKE_SA established: Main tunnel is active and authenticated
- CHILD_SA INSTALLED: Data traffic encryption is active with specific SPIs (Security Parameter Indexes)
- TUNNEL mode: Packets are encapsulated and encrypted for inter-site communication
- DPD active: Dead peer detection monitors connection health

5. IPSec Tunnel Establishment and ESP Traffic Verification

5.1 Verification Of IPSec



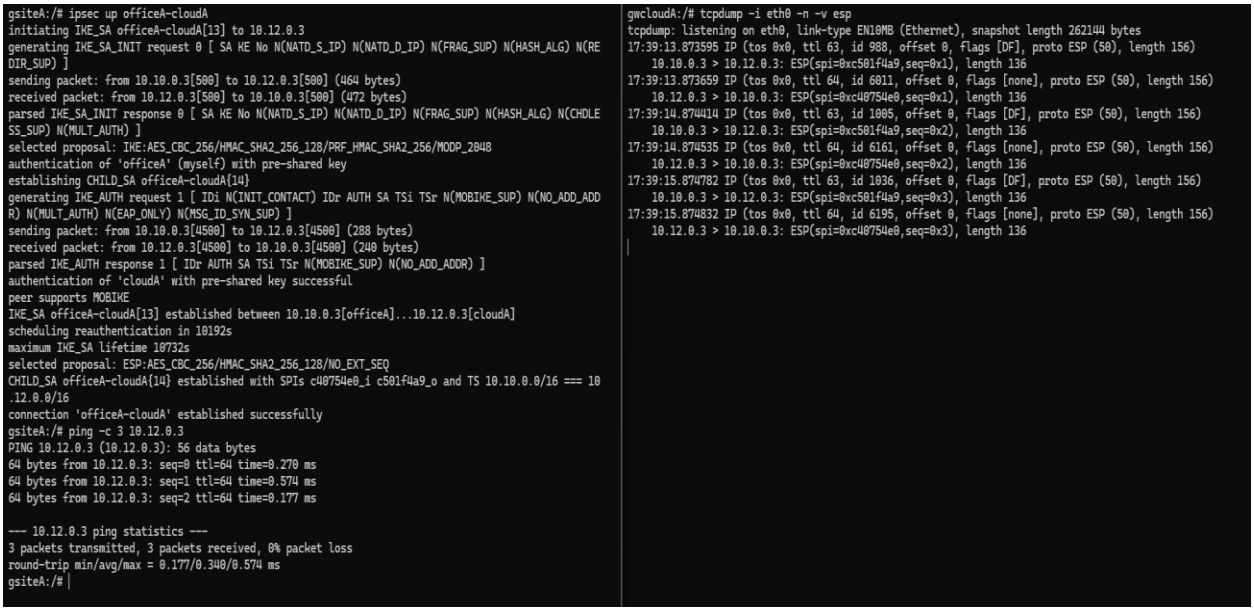
- On the right, tcpdump is running to capture IKE and ESP packets (tcpdump -i eth0 -n udp port 500 or udp port 4500).
- The output shows expected ISAKMP exchanges:
  - ❖ IKE_SA_INIT (port 500: parent_sa ikev2_init[I]/[R])

- ❖ IKE_AUTH, followed by CHILD_SA creation and information exchange (port 4500, NONESP-encap).

Result: Live capture matches the IPSec key exchange process: gsiteA's initiation, negotiation, key exchange, and deletion are all visible on gwcloudA, confirming both communication and security association management.

5.2 ESP Traffic Verification

After establishing the tunnel and generating traffic, tcpdump was run on the cloud gateway while a connection (ping) was initiated from OfficeA to CloudA. The screenshot below documents encrypted ESP packets exchanged between 10.10.0.3 (OfficeA gateway) and 10.12.0.3 (CloudA gateway):



This confirms that traffic is being encrypted according to the IPSec ESP protocol and that secure tunnel operation is in place as required by the lab objectives.

6. Conclusion

Conclusion The implementation of IPSec tunnels between Office Sites A and B and Cloud Site A has been successfully completed. Both tunnels are actively encrypting traffic between internal networks using military-grade cryptography (AES-256) and modern protocol standards (IKEv2).

The tunnels provide:

- Confidentiality: All data encrypted end-to-end using AES-256.
- Integrity: HMAC-SHA256 ensures data has not been modified.
- Authentication: Pre-shared keys verify gateway identities.
- Availability: Dead Peer Detection maintains tunnel resilience.

All configuration files, tunnel establishment commands, and status outputs have been documented and exported for audit and compliance purposes. The infrastructure is ready for production use and meets security requirements for sensitive inter-site communication.