

# Advanced Network Security and Architectures

*Laboratory Report:*

*Module 2*



***Group 9***

Félix Saraiva – 98752

Rafael Martins – 90629

Miguel Ribeiro – 87553

## Table of Contents:

Table of Figures:.....	3
1. Lab No. 2.1 – IPsec and VPNs .....	6
1.1. IPsec using ESP in tunnel mode .....	6
Network Architecture: .....	7
Proof-of-Concept: .....	7
1.2. IPsec with digital certificates and certificate authority .....	14
Network Architecture: .....	14
Proof-of-Concept: .....	14
1.3. IPsec with NAT traversal .....	20
Network Architecture: .....	20
Proof-of-Concept: .....	20
1.4. GRE over IPsec.....	26
Network Architecture: .....	26
Proof-of-Concept: .....	26
1.5. DMVPN Phase 3.....	32
Network Architecture: .....	32
Proof-of-Concept: .....	33
1.6. DMVPN over IPsec .....	37
Network Architecture: .....	37
Proof-of-Concept: .....	37
2. Lab No. 2.2 – Networking of virtual containers.....	41
2.1. Connecting network namespaces to external networks .....	41
Network Architecture: .....	41
Proof-of-Concept: .....	41
2.2. Macvlan networks with VLANs.....	43
Network Architecture: .....	43
Proof-of-Concept: .....	43
2.3. Linux VxLAN with multicast routing .....	46
Network Architecture: .....	46
Proof-of-Concept: .....	47

2.4. Docker Swarm without data encryption .....	51
Network Architecture: .....	51
Proof-of-Concept: .....	52
2.5. Docker Swarm with data encryption.....	54
Proof-of-Concept: .....	54
References: .....	56
Annex A: Configurations .....	57
IPSec using ESP in tunnel mode .....	57
IPSec with digital certificates and certificate authority.....	59
IPSec with NAT traversal .....	62
GRE over IPSec .....	64
DMVPN Phase 3 .....	67
DMVPN over IPSec .....	70
Annex B: Configurations .....	74
Connecting network namespaces to external networks .....	74
Macvlan networks with VLANs .....	75
Linux VxLAN with multicast routing.....	77
Docker Swarm without data encryption.....	79
Docker Swarm with data encryption .....	82

## Table of Figures:

Figure 1 - IPSec using ESP tunnel mode Network Architecture.....	7
Figure 2 - ISAKMP messages .....	7
Figure 3 - First ISAKMP message.....	8
Figure 4 - ISAKMP response from R2 .....	9
Figure 5 - Key exchange message 3 .....	10
Figure 6 - Key exchange message 4 .....	10
Figure 7 - ISAKMP Authentication of peers .....	11
Figure 8 - Three quick mode messages.....	12
Figure 9 - ESP pings from PC2 to PC1.....	12
Figure 10 - Informational ISAKMP message NO-PROPOSAL-CHOSEN .....	13
Figure 11 - IPSec with digital certificates Network Architecture.....	14
Figure 12 - Information of the PKI server .....	14
Figure 13 - CA certificate Information .....	15
Figure 14 - Public key created for the CA .....	15
Figure 15 - SCEP HTTP messages between R1 and RA.....	15
Figure 16 - R1's Public Key .....	15
Figure 17 - Certificate installed at R1.....	15
Figure 18 - GET certificate HTTP message .....	16
Figure 19 - HTTP OK message containing the CA certificate .....	16
Figure 20 - Certificate Details.....	17
Figure 21 - Certificate General Info .....	17
Figure 22 - Ping from PC1 to PC2 .....	17
Figure 23 - IKE authentication method using Digital certificates .....	18
Figure 24 - R2 Public key at stored in R1 .....	19
Figure 25 – Cyphered ICMP messages between R1 and R2 .....	19
Figure 26 - IPSec with NAT traversal Network Architecture.....	20
Figure 27 - Packet capture between R1 and RA (Private link) .....	21
Figure 28 - Packet capture on the public link .....	21
Figure 29 - Third ISAKMP message with NAT-D Payload .....	22
Figure 30 - Fourth ISAKMP message with NAT-D Payload.....	23
Figure 31 - Fifth ISAKMP packet with port 4500.....	24
Figure 32 - ESP packets in the private network .....	25
Figure 33 - ESP packets in the public network.....	25
Figure 34 - GRE over IPSec Network Architecture.....	26
Figure 35 - ICMP Packet from PC1 to PC2 (AH Tunnel mode) .....	26
Figure 36 - GRE over IPSec AH Tunnel Mode packet structure .....	27
Figure 37 - OSPF Hello Packet Public network.....	27
Figure 38 - OSPF Hello Packet Private network .....	28
Figure 39 - Router 2 OSPF LSDB Private network .....	29

Figure 40 - R1 to RA connection using ESP tunnel mode .....	30
Figure 41 - OSPF Hello Public Network transport mode.....	30
Figure 42 - ICMP packet PC1 to PC2 AH transport mode .....	31
Figure 43 - OSPF packet Private Network AH transport mode.....	31
Figure 44 - DMVPN Network Architecture .....	32
Figure 45 - RIPv2 packet on the public network.....	33
Figure 46 - OSPF packet on the public network.....	33
Figure 47 - RIPv2 packet with no ip summary-address .....	34
Figure 48 - R1's Routing table .....	34
Figure 49 – RIPv2 packet with no Split-horizon and no summary-address.....	35
Figure 50 - NHRP registration request packet .....	35
Figure 51 - R1's NHRP cache .....	36
Figure 52 - DMVPN over IPsec Network Architecture.....	37
Figure 53 - R1's routing table.....	38
Figure 54 - R1's NHRP cache .....	38
Figure 55 - NHRP Registration Reply with IPsec AH transport mode .....	38
Figure 56 - RIPv2 Response packet with IPsec AH transport mode .....	39
Figure 57 - OSPF Hello packet with no IPsec protection .....	39
Figure 58 - Capture of packets transmitted with IPsec ESP transport mode .....	39
Figure 59 – R1's SAs in DMVPN over IPsec.....	40
Figure 60 - Network namespaces connecting to external networks network architecture.....	41
Figure 61 - Active NNSes.....	41
Figure 62 - Bridge, red and blue interfaces .....	42
Figure 63 - Successful communication with external networks .....	42
Figure 64 - Macvlan networks with VLANs Network Architecture .....	43
Figure 65 - ub1 docker networks .....	44
Figure 66 - Sub-interfaces in ub1 .....	44
Figure 67 - Containers in ub1.....	44
Figure 68 - Containers in ub2.....	44
Figure 69 - Ping from cont0 to cont2 (VLAN1).....	45
Figure 70 - Ping from cont1 to cont3 (VLAN2).....	45
Figure 71 - Ping from cont0 to cont3 (VLAN1 to VLAN2).....	45
Figure 72 - ICMP packet between VLAN1 containers .....	45
Figure 73 - ICMP packet between VLAN2 containers .....	46
Figure 74 - Linux VxLAN with multicast routing Network Architecture .....	46
Figure 75 - IGMP Groups R1.....	47
Figure 76 - Membership request IGMP messages of UB1 and UB2 .....	47
Figure 77 - Membership request IGMP messages of UB3 .....	48
Figure 78 - R1's Multicast routing table.....	49
Figure 79 - ICMP ping from b1 to b2.....	49
Figure 80 - ICMP ping from r1 to b3 .....	49

Figure 81 – ARP reply message from UB2 to UB1 .....	50
Figure 82 - ICMP packet from r1 to r4 .....	50
Figure 83 - ARP cache of r1 .....	50
Figure 84 - Docker Swarm without data encryption network architecture .....	51
Figure 85 - Docker swarm node's roles .....	52
Figure 86 - Connectivity test between all the containers.....	52
Figure 87 - Blue network containers of UB1 .....	53
Figure 88 - Blue network VxLAN ID.....	53
Figure 89 - ICMP packet from b1 to b5.....	53
Figure 90 - Cyphered ICMP packet from b1 to b5 .....	54
Figure 91 - Encrypted entry at blue network UB1 .....	55

## 1. Lab No. 2.1 – IPSec and VPNs

This laboratory work aims to study IPSec and VPNs. The laboratories analysed are:

- I. IPSec using ESP in tunnel mode
- II. IPSec with digital certificates and certificate authority
- III. IPSec with NAT traversal
- IV. GRE over IPSec
- V. DMVPN Phase 3
- VI. DMVPN over IPSec

### 1.1. IPSec using ESP in tunnel mode

In this exercise, we connect two branches of an organisation using an IPSec tunnel. After the address configurations, we have configured **OSPF** as routing protocol in the public subnets and static routes for the organisation to be able to communicate externally. The complete configurations are present in [Annex A](#).

IPSec can be used in several situations, such as establishing a secure link between a firewall and a host. Knowing the requirements in terms of configuration to establish an IPSec tunnel is essential.

So, to make IPSec work through your firewalls, you should open UDP port 500 and permit IP protocol numbers 50 and 51 on both inbound and outbound firewall filters. UDP Port 500 should be opened to allow Internet Security Association and Key Management Protocol (ISAKMP) traffic to be forwarded through your firewalls. IP protocol ID 50 should be set to enable IPSec Encapsulating Security Protocol (ESP) traffic to be delivered. Finally, IP protocol ID 51 should be set to allow Authentication Header (AH) traffic to be forwarded. (Clercq, 2012)

## Network Architecture:

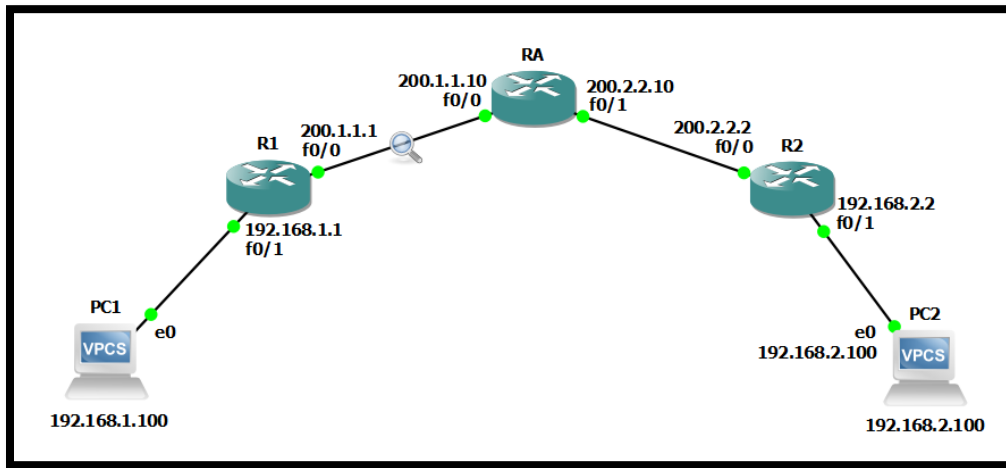


Figure 1 - IPsec using ESP tunnel mode Network Architecture

Regarding the IPsec configuration, we followed the configuration provided for R1 and adapted it for R2, changing the pre-shared key IP, the peer, and the access list. To better analyse the establishment of the IPsec tunnel, we configured a Wireshark capture between R1 and RA.

## Proof-of-Concept:

1. The Tunnel establishment involves **IKE Phase 1** and **IKE Phase 2**. Both IKE phases use the **ISAKMP (Internet Security Association and Key Management Protocol)**. IKE Phase 1 consists of six messages in **Main Mode**, and IKE Phase 2 consists of 3 messages in **Quick Mode**, as shown in Figure 2.

No.	Time	Source	Destination	Protocol	Length	Info
127	260.583370	200.1.1.1	200.2.2.2	ISAKMP	210	Identity Protection (Main Mode)
129	260.768275	200.2.2.2	200.1.1.1	ISAKMP	150	Identity Protection (Main Mode)
130	260.954014	200.1.1.1	200.2.2.2	ISAKMP	390	Identity Protection (Main Mode)
131	261.203585	200.2.2.2	200.1.1.1	ISAKMP	410	Identity Protection (Main Mode)
132	261.453146	200.1.1.1	200.2.2.2	ISAKMP	150	Identity Protection (Main Mode)
133	261.515833	200.2.2.2	200.1.1.1	ISAKMP	118	Identity Protection (Main Mode)
134	261.566254	200.1.1.1	200.2.2.2	ISAKMP	230	Quick Mode
135	261.629469	200.2.2.2	200.1.1.1	ISAKMP	230	Quick Mode
136	261.657658	200.1.1.1	200.2.2.2	ISAKMP	102	Quick Mode

Figure 2 - ISAKMP messages

2. In Figure 3, we can see the first message exchange between peers. This message contains the cryptography algorithms supported by R1 to start the **Security Association (SA)** negotiation with R2. In the packet information identified by "**IKE Attribute**", we can observe the protocols to be used.



isakmp						
No.	Time	Source	Destination	Protocol	Length	Info
127	260.583370	200.1.1.1	200.2.2.2	ISAKMP	210	Identity Protection (Main Mode)
> Frame 127: 210 bytes on wire (1680 bits), 210 bytes captured (1680 bits) on interface -, id 0 > Ethernet II, Src: ca:01:07:ed:00:08 (ca:01:07:ed:00:08), Dst: c2:03:08:12:00:00 (c2:03:08:12:00:00) > Internet Protocol Version 4, Src: 200.1.1.1, Dst: 200.2.2.2 > User Datagram Protocol, Src Port: 500, Dst Port: 500 ✓ Internet Security Association and Key Management Protocol						
Initiator SPI: e8a8c78a3c23cf36 Responder SPI: 0000000000000000 Next payload: Security Association (1) > Version: 1.0 Exchange type: Identity Protection (Main Mode) (2) > Flags: 0x00 Message ID: 0x00000000 Length: 168						
✓ Payload: Security Association (1) Next payload: Vendor ID (13) Reserved: 00 Payload length: 60 Domain of interpretation: IPSEC (1) > Situation: 00000001						
✓ Payload: Proposal (2) # 1 Next payload: NONE / No Next Payload (0) Reserved: 00 Payload length: 48 Proposal number: 1 Protocol ID: ISAKMP (1) SPI Size: 0 Proposal transforms: 1						
✓ Payload: Transform (3) # 1 Next payload: NONE / No Next Payload (0) Reserved: 00 Payload length: 40 Transform number: 1 Transform ID: KEY_IKE (1) Reserved: 0000 > IKE Attribute (t=1,l=2): Encryption-Algorithm: AES-CBC > IKE Attribute (t=14,l=2): Key-Length: 256 > IKE Attribute (t=2,l=2): Hash-Algorithm: SHA > IKE Attribute (t=4,l=2): Group-Description: 1536 bit MODP group > IKE Attribute (t=3,l=2): Authentication-Method: Pre-shared key > IKE Attribute (t=11,l=2): Life-Type: Seconds > IKE Attribute (t=12,l=4): Life-Duration: 86400						
> Payload: Vendor ID (13) : RFC 3947 Negotiation of NAT-Traversal in the IKE > Payload: Vendor ID (13) : draft-ietf-ipsec-nat-t-ike-07 > Payload: Vendor ID (13) : draft-ietf-ipsec-nat-t-ike-03 > Payload: Vendor ID (13) : draft-ietf-ipsec-nat-t-ike-02\n						

Figure 3 - First ISAKMP message

3. In Figure 4, we have the response from R2. In this packet, we have the confirmation from R2 of the cryptography algorithms to be used in the SA.

isakmp						
No.	Time	Source	Destination	Protocol	Length	Info
127	260.583370	200.1.1.1	200.2.2.2	ISAKMP	210	Identity Protection (Main Mode)
129	260.768275	200.2.2.2	200.1.1.1	ISAKMP	150	Identity Protection (Main Mode)

```

> Frame 129: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits) on interface -, id 0
> Ethernet II, Src: c2:03:08:12:00:00 (c2:03:08:12:00:00), Dst: ca:01:07:ed:00:08 (ca:01:07:ed:00:08)
> Internet Protocol Version 4, Src: 200.2.2.2, Dst: 200.1.1.1
> User Datagram Protocol, Src Port: 500, Dst Port: 500
v Internet Security Association and Key Management Protocol
  Initiator SPI: e8a8c78a3c23cf36
  Responder SPI: fe069fbcaa3cfdb7
  Next payload: Security Association (1)
  > Version: 1.0
  > Exchange type: Identity Protection (Main Mode) (2)
  > Flags: 0x00
  > Message ID: 0x00000000
  > Length: 108
v Payload: Security Association (1)
  Next payload: Vendor ID (13)
  > Reserved: 00
  > Payload length: 60
  > Domain of interpretation: IPSEC (1)
  > Situation: 00000001
v Payload: Proposal (2) # 1
  Next payload: NONE / No Next Payload (0)
  > Reserved: 00
  > Payload length: 48
  > Proposal number: 1
  > Protocol ID: ISAKMP (1)
  > SPI Size: 0
  > Proposal transforms: 1
v Payload: Transform (3) # 1
  Next payload: NONE / No Next Payload (0)
  > Reserved: 00
  > Payload length: 40
  > Transform number: 1
  > Transform ID: KEY_IKE (1)
  > Reserved: 0000
  > IKE Attribute (t=1,l=2): Encryption-Algorithm: AES-CBC
  > IKE Attribute (t=14,l=2): Key-Length: 256
  > IKE Attribute (t=2,l=2): Hash-Algorithm: SHA
  > IKE Attribute (t=4,l=2): Group-Description: 1536 bit MODP group
  > IKE Attribute (t=3,l=2): Authentication-Method: Pre-shared key
  > IKE Attribute (t=11,l=2): Life-Type: Seconds
  > IKE Attribute (t=12,l=4): Life-Duration: 86400
v Payload: Vendor ID (13) : RFC 3947 Negotiation of NAT-Traversal in the IKE
  Next payload: NONE / No Next Payload (0)
  > Reserved: 00
  > Payload length: 20
  > Vendor ID: 4a131c81070358455c5728f20e95452f
  > Vendor ID: RFC 3947 Negotiation of NAT-Traversal in the IKE

```

Figure 4 - ISAKMP response from R2

4. The first phase is now completed. Secondly, both parts must compute the shared key using the **Diffie-Helman (DH)** algorithm based on the **DH** group previously agreed (DH group 5 in this case). DH allows both peers to agree on a shared key. We can see this process in the packets in Figure 5 and Figure 6.

No.	Time	Source	Destination	Protocol	Length	Info
127	260.583370	200.1.1.1	200.2.2.2	ISAKMP	210	Identity Protection (Main Mode)
129	260.768275	200.2.2.2	200.1.1.1	ISAKMP	150	Identity Protection (Main Mode)
130	260.954014	200.1.1.1	200.2.2.2	ISAKMP	390	Identity Protection (Main Mode)

> Frame 130: 390 bytes on wire (3120 bits), 390 bytes captured (3120 bits) on interface -, id 0 > Ethernet II, Src: ca:01:07:ed:00:08 (ca:01:07:ed:00:08), Dst: c2:03:08:12:00:00 (c2:03:08:12:00:00) > Internet Protocol Version 4, Src: 200.1.1.1, Dst: 200.2.2.2 > User Datagram Protocol, Src Port: 500, Dst Port: 500 > Internet Security Association and Key Management Protocol Initiator SPI: e8a8c78a3c23cf36 Responder SPI: fe069fbcaa3cfdb7 Next payload: Key Exchange (4) > Version: 1.0 Exchange type: Identity Protection (Main Mode) (2) > Flags: 0x00 Message ID: 0x00000000 Length: 348 > Payload: Key Exchange (4) Next payload: Nonce (10) Reserved: 00 Payload length: 196 Key Exchange Data: 54262db48445bda1e1942d567c36ad96cc35951583cc90e0c88ceecd8587944c369c4857... > Payload: Nonce (10) Next payload: Vendor ID (13) Reserved: 00 Payload length: 24 Nonce DATA: 304996faeaaafca61fd1887de42f88b1f34db88a
---

Figure 5 - Key exchange message 3

No.	Time	Source	Destination	Protocol	Length	Info
127	260.583370	200.1.1.1	200.2.2.2	ISAKMP	210	Identity Protection (Main Mode)
129	260.768275	200.2.2.2	200.1.1.1	ISAKMP	150	Identity Protection (Main Mode)
130	260.954014	200.1.1.1	200.2.2.2	ISAKMP	390	Identity Protection (Main Mode)
131	261.203585	200.2.2.2	200.1.1.1	ISAKMP	410	Identity Protection (Main Mode)

> Frame 131: 410 bytes on wire (3280 bits), 410 bytes captured (3280 bits) on interface -, id 0 > Ethernet II, Src: c2:03:08:12:00:00 (c2:03:08:12:00:00), Dst: ca:01:07:ed:00:08 (ca:01:07:ed:00:08) > Internet Protocol Version 4, Src: 200.2.2.2, Dst: 200.1.1.1 > User Datagram Protocol, Src Port: 500, Dst Port: 500 > Internet Security Association and Key Management Protocol Initiator SPI: e8a8c78a3c23cf36 Responder SPI: fe069fbcaa3cfdb7 Next payload: Key Exchange (4) > Version: 1.0 Exchange type: Identity Protection (Main Mode) (2) > Flags: 0x00 Message ID: 0x00000000 Length: 368 > Payload: Key Exchange (4) Next payload: Nonce (10) Reserved: 00 Payload length: 196 Key Exchange Data: faf4e507d3c1253ee0fb2249216f91ef95eae06be2ff325472e18b15adef86422d58a97f... > Payload: Nonce (10) Next payload: Vendor ID (13) Reserved: 00 Payload length: 24 Nonce DATA: a1e6741baea2254a57e5ad1e21262d15732ec08d
---

Figure 6 - Key exchange message 4

5. In both packets, the payload Key Exchange contains the value computed from each peer that will be used later in the DH process by the other end to compute the Shared Key. Some literature calls the Diffie-Hellman algorithm a Key Exchange protocol. We can see that this can be, in a way, incoherent.

The DH protocol is a Public Key Generation Protocol, where each party shares a parameter with the other end for both ends to compute a shared secret from those parameters. What is shared is merely a component that generates the shared key, not the key itself.

6. We have now finished the second step of IKE Phase 1.

As the last of IKE Phase 1, we have authentication of peers using the agreed authentication method. This is done by exchanging hash values between peers, as seen in Figure 7.

isakmp						
No.	Time	Source	Destination	Protocol	Length	Info
127	260.583370	200.1.1.1	200.2.2.2	ISAKMP	210	Identity Protection (Main Mode)
129	260.768275	200.2.2.2	200.1.1.1	ISAKMP	150	Identity Protection (Main Mode)
130	260.954014	200.1.1.1	200.2.2.2	ISAKMP	390	Identity Protection (Main Mode)
131	261.203585	200.2.2.2	200.1.1.1	ISAKMP	410	Identity Protection (Main Mode)
132	261.453146	200.1.1.1	200.2.2.2	ISAKMP	150	Identity Protection (Main Mode)

>	Frame 132: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits) on interface -, id 0
>	Ethernet II, Src: ca:01:07:ed:00:08 (ca:01:07:ed:00:08), Dst: c2:03:08:12:00:00 (c2:03:08:12:00:00)
>	Internet Protocol Version 4, Src: 200.1.1.1, Dst: 200.2.2.2
>	User Datagram Protocol, Src Port: 500, Dst Port: 500
>	Internet Security Association and Key Management Protocol
	Initiator SPI: e8a8c78a3c23cf36
	Responder SPI: fe069fbcaa3cfd7b
	Next payload: Identification (5)
>	Version: 1.0
	Exchange type: Identity Protection (Main Mode) (2)
>	Flags: 0x01
	Message ID: 0x00000000
	Length: 108
	Encrypted Data (80 bytes)

Figure 7 - ISAKMP Authentication of peers

7. Moving on to **IKE Phase 2**, the tunnel created is used to protect data. We have three messages in this phase, as shown in Figure 8.

isakmp						
No.	Time	Source	Destination	Protocol	Length	Info
127	260.583370	200.1.1.1	200.2.2.2	ISAKMP	210	Identity Protection (Main Mode)
129	260.768275	200.2.2.2	200.1.1.1	ISAKMP	150	Identity Protection (Main Mode)
130	260.954014	200.1.1.1	200.2.2.2	ISAKMP	390	Identity Protection (Main Mode)
131	261.203585	200.2.2.2	200.1.1.1	ISAKMP	410	Identity Protection (Main Mode)
132	261.453146	200.1.1.1	200.2.2.2	ISAKMP	150	Identity Protection (Main Mode)
133	261.515833	200.2.2.2	200.1.1.1	ISAKMP	118	Identity Protection (Main Mode)
134	261.566254	200.1.1.1	200.2.2.2	ISAKMP	230	Quick Mode
135	261.629469	200.2.2.2	200.1.1.1	ISAKMP	230	Quick Mode
136	261.657658	200.1.1.1	200.2.2.2	ISAKMP	102	Quick Mode

>	Frame 134: 230 bytes on wire (1840 bits), 230 bytes captured (1840 bits) on interface -, id 0					
>	Ethernet II, Src: ca:01:07:ed:00:08 (ca:01:07:ed:00:08), Dst: c2:03:08:12:00:00 (c2:03:08:12:00:00)					
>	Internet Protocol Version 4, Src: 200.1.1.1, Dst: 200.2.2.2					
>	User Datagram Protocol, Src Port: 500, Dst Port: 500					
>	Internet Security Association and Key Management Protocol					
	Initiator SPI: e8a8c78a3c23cf36					
	Responder SPI: fe069fbcaa3cfdb7					
	Next payload: Hash (8)					
>	Version: 1.0					
	Exchange type: Quick Mode (32)					
>	Flags: 0x01					
	Message ID: 0xf56a8d81					
	Length: 188					
	Encrypted Data (160 bytes)					

Figure 8 - Three quick mode messages

8. After both phases are completed, it is expected to see the packets exchanged between PC1 and PC2 using the ESP protocol. We can verify this in Figure 9, where we ping PC2 from PC1.

esp						
No.	Time	Source	Destination	Protocol	Length	Info
137	262.500154	200.1.1.1	200.2.2.2	ESP	166	ESP (SPI=0xeb19d137)
138	262.647960	200.2.2.2	200.1.1.1	ESP	166	ESP (SPI=0x19e39ba9)
139	263.659496	200.1.1.1	200.2.2.2	ESP	166	ESP (SPI=0xeb19d137)
140	263.701495	200.2.2.2	200.1.1.1	ESP	166	ESP (SPI=0x19e39ba9)
141	264.725651	200.1.1.1	200.2.2.2	ESP	166	ESP (SPI=0xeb19d137)
142	264.768415	200.2.2.2	200.1.1.1	ESP	166	ESP (SPI=0x19e39ba9)
144	265.783306	200.1.1.1	200.2.2.2	ESP	166	ESP (SPI=0xeb19d137)
145	265.825681	200.2.2.2	200.1.1.1	ESP	166	ESP (SPI=0x19e39ba9)

>	Frame 137: 166 bytes on wire (1328 bits), 166 bytes captured (1328 bits) on interface -, id 0					
>	Ethernet II, Src: ca:01:07:ed:00:08 (ca:01:07:ed:00:08), Dst: c2:03:08:12:00:00 (c2:03:08:12:00:00)					
>	Internet Protocol Version 4, Src: 200.1.1.1, Dst: 200.2.2.2					
>	Encapsulating Security Payload					
	ESP SPI: 0xeb19d137 (3944337719)					
	ESP Sequence: 1					

Figure 9 - ESP pings from PC2 to PC1

9. Since the traffic is cyphered with the shared key established in IKE Phase 1, we cannot recognise the ICMP packets originating from the ping. We can see that the source and destination of the packets correspond to the routers at the tunnel's beginning and end.

Additionally, we can see that a new IP header was added by observing the payload. These are evidence that the tunnel is using ESP in tunnel mode.

10. Using the different ISAKMP policies presented below, at R2 (encryption type DES), we concluded that the ISAKMP SA negotiation with R2 has failed, thanks to a mismatch of the proposed encryption protocols. In this case, a NO-PROPOSAL-CHOSEN message is sent, as shown in Figure 10.

```
1. #Different Policy
2. #IKE Phase 1 - Configure ISAKMP policy
3. crypto isakmp policy 10
4. hash md5
5. authentication pre-share
6. group 5
7. lifetime 86400
8. encryption des
```

isakmp						
No.	Time	Source	Destination	Protocol	Length	Info
11	19.700678	200.1.1.1	200.2.2.2	ISAKMP	210	Identity Protection (Main Mode)
12	19.919424	200.2.2.2	200.1.1.1	ISAKMP	142	Informational
17	29.450986	200.1.1.1	200.2.2.2	ISAKMP	210	Identity Protection (Main Mode)
22	39.451643	200.1.1.1	200.2.2.2	ISAKMP	210	Identity Protection (Main Mode)
27	49.454407	200.1.1.1	200.2.2.2	ISAKMP	210	Identity Protection (Main Mode)
32	59.455193	200.1.1.1	200.2.2.2	ISAKMP	210	Identity Protection (Main Mode)
38	69.458782	200.1.1.1	200.2.2.2	ISAKMP	210	Identity Protection (Main Mode)

> Frame 12: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits) on interface -, id 0 > Ethernet II, Src: c2:03:08:12:00:00 (c2:03:08:12:00:00), Dst: ca:01:07:ed:00:08 (ca:01:07:ed:00:08) > Internet Protocol Version 4, Src: 200.2.2.2, Dst: 200.1.1.1 > User Datagram Protocol, Src Port: 500, Dst Port: 500 > Internet Security Association and Key Management Protocol						
Initiator SPI: f0d0842d6139739e Responder SPI: 4a33eab92dd4618c Next payload: Notification (11)						
> Version: 1.0 Exchange type: Informational (5)						
> Flags: 0x00 Message ID: 0x00000000 Length: 100						
> Payload: Notification (11)						
Next payload: NONE / No Next Payload (0) Reserved: 00 Payload length: 72 Domain of interpretation: IPSEC (1) Protocol ID: ISAKMP (1) SPI Size: 0						
Notify Message Type: NO-PROPOSAL-CHOSEN (14)						
Notification DATA: 0d00003c000000010000000100000000644a2bc4695632c00000000065d7112800000a1c...						

Figure 10 - Informational ISAKMP message NO-PROPOSAL-CHOSEN

## 1.2. IPSec with digital certificates and certificate authority

In this exercise, we configure a Certification Authority (CA) and IPSec using digital certificates. Authentication in IPSec can be provided through pre-shared keys or digital certificates, which requires a PKI infrastructure, in this case, a CA Server, trusted by both parties. Although more complex, using digital certificates provides a higher level of scalability.

These certificates are tamper-proof and cannot be forged. The most widely used format for digital certificates is X.509. The complete configuration can be found in [Annex A](#).

### Network Architecture:

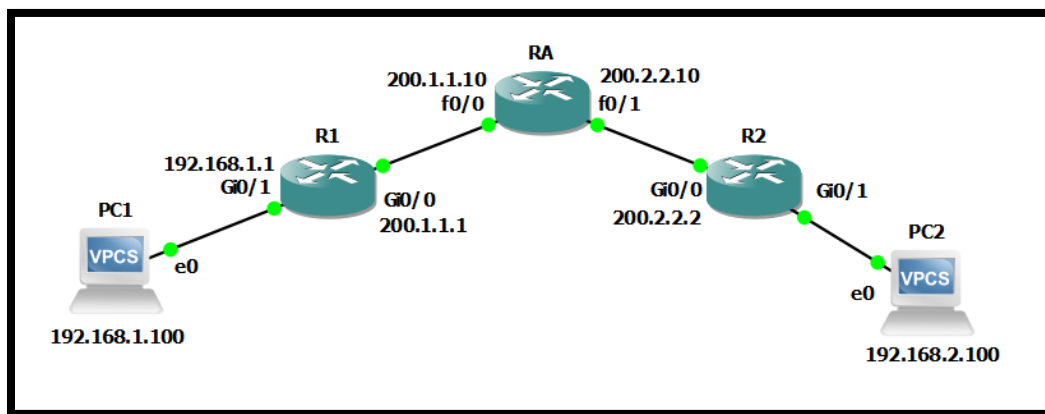


Figure 11 - IPSec with digital certificates Network Architecture

This architecture applies two IOSv routers, R1 and R2, and a 7200 router as RA, which serves as the CA.

### Proof-of-Concept:

1. After assigning all the IP addresses specified in Figure 11 and configuring OSPF with the respective static routes, we configure the CA at router RA. The information on the PKI server can be seen in Figure 12. In Figure 13, we display the information of the CA certificate, and in Figure 14, we can see the public key created for the CA.

```
RA#show crypto pki server
Certificate Server myCA:
  Status: enabled
  State: enabled
  Server's configuration is locked (enter "shut" to unlock it)
  Issuer name: cn=saarCA
  CA cert fingerprint: 07C75185 EBD04A45 A413EA43 A290747C
  Granting mode is: auto
  Last certificate issued serial number (hex): 1
  CA certificate expiration timer: 17:28:28 UTC Jun 5 2025
  CRL NextUpdate timer: 23:28:28 UTC Jun 6 2022
  Current primary storage dir: nvram:
  Database Level: Minimum - no cert data written to storage
```

Figure 12 - Information of the PKI server



```

RA#show crypto ca certificate
CA Certificate
  Status: Available
  Certificate Serial Number (hex): 01
  Certificate Usage: Signature
  Issuer:
    cn=saarCA
  Subject:
    cn=saarCA
  Validity Date:
    start date: 17:28:28 UTC Jun 6 2022
    end   date: 17:28:28 UTC Jun 5 2025
  Associated Trustpoints: myCA

```

Figure 13 - CA certificate Information

```

RA#show crypto key mypub rsa
% Key pair was generated at: 17:28:28 UTC Jun 6 2022
Key name: myCA
Key type: RSA KEYS
Storage Device: not specified
Usage: General Purpose Key
Key is not exportable.
Key Data:
  30819F30 0D06092A 864886F7 0D010101 05000381 8D003081 89028181 00B2C86A
  264C6729 2A9D85FD 12A31886 41C4FDD0 0F9513CE 1F523173 83D83FD8 C4C90B43
  68EF1B7A 21E8210F 0667A296 D50403EF 4E62B037 0EFAE409 F8D74469 F70C1226
  FF0D0242 E5A5B078 A86FE618 68CE44D3 435F26C6 8DCE99C8 42F614FF 70D3E705
  2B750669 DF2C3D24 EBA9FE2C A214694C 8814E9E3 8F911369 AE62742E 07020301 0001
% Key pair was generated at: 17:28:29 UTC Jun 6 2022
Key name: myCA.server
Key type: RSA KEYS
Temporary key
Usage: Encryption Key
Key is not exportable.
Key Data:
  307C300D 06092A86 4886F70D 01010105 00036B00 30680261 00AE5D46 1DC51415
  A3E4AD9E BAD21CE0 DF449F71 A286175E 30EF7CA6 D3A5EF84 031E1998 D636EC69
  CDABA73C 5E9C3B59 34698AA5 F1C8FC07 30A1ADE4 4EEF43AA 9292A5F3 30347123
  756E774E FD09D25E B8896E54 CE65AEDC 62F3C54E 2271953B B8020301 0001
RA#

```

Figure 14 - Public key created for the CA

- We now configure R1 and R2 as the PKI clients who obtain a CA certificate. We used a (Simple Certificate Enrolment Protocol) SCEP-based enrolment. The exchange of HTTP messages related to SCEP can be seen in Figure 15. We can also confirm that the certificate was installed at R1 in Figure 17 and see R1's public key in Figure 16.

No.	Time	Source	Destination	Protocol	Length	Info
42	84.665389	200.1.1.1	200.1.1.10	HTTP	211	GET /cgi-bin/pkiclient.exe?operation=GetCACert&message=myTrustpoint HTTP/1.0
45	84.729123	200.1.1.10	200.1.1.1	HTTP	603	HTTP/1.1 200 OK (application/x-x509-ca-cert)
52	84.775835	200.1.1.1	200.1.1.10	HTTP	211	GET /cgi-bin/pkiclient.exe?operation=GetCACaps&message=myTrustpoint HTTP/1.0
55	84.791110	200.1.1.10	200.1.1.1	HTTP	112	HTTP/1.1 200 OK (application/x-pki-message)
81	119.343694	200.1.1.1	200.1.1.10	HTTP	901	GET /cgi-bin/pkiclient.exe?operation=PKIOperation&message=MIIF8gY3KoZiHvcNAQCoC
86	119.571840	200.1.1.10	200.1.1.1	HTTP	166	Continuation

Figure 15 - SCEP HTTP messages between R1 and RA

```

Certificate
  Status: Available
  Certificate Serial Number (hex): 02
  Certificate Usage: General Purpose
  Issuer:
    cn=saarCA
  Subject:
    Name: Router
    Serial Number: 9VILXPB82TVPLFP1BDZQB
    hostname=Router+serialNumber=9VILXPB82TVPLFP1BDZQB
  Validity Date:
    start date: 17:40:39 UTC Jun 6 2022
    end   date: 17:40:39 UTC Jun 6 2023
  Associated Trustpoints: myTrustpoint

CA Certificate
  Status: Available
  Certificate Serial Number (hex): 01
  Certificate Usage: Signature
  Issuer:
    cn=saarCA
  Subject:
    cn=saarCA
  Validity Date:
    start date: 17:28:28 UTC Jun 6 2022
    end   date: 17:28:28 UTC Jun 5 2025
  Associated Trustpoints: myTrustpoint

```

Figure 17 - Certificate installed at R1

```

Router(config)#do show crypto key mypub rsa
% Key pair was generated at: 17:39:58 UTC Jun 6 2022
Key name: Router
Key type: RSA KEYS
Storage Device: not specified
Usage: General Purpose Key
Key is not exportable.
Key Data:
  30819F30 0D06092A 864886F7 0D010101 05000381 8D003081 89028181 00BD495D
  D084C18D 1F4A8972 23205488 9666B653 D00E702B E7CF00FB D6942C79 E621DDCE
  3880D0A0 D4C2220D 05532DAB 440C8277 27B6CD32 38188CDB F79BDC77 B54A1DF2
  FB5E5A4E B0500FC6 157D9F82 676D00F8 840A5D5A 2F2CC115 1CF86E6E 02031527
  9F8E6CDA 5477F4FA 476D2D0E 4505E852 0D9C38BD 45B214B1 88E6EEBE BF020301 0001
% Key pair was generated at: 17:40:00 UTC Jun 6 2022
Key name: Router.server
Key type: RSA KEYS
Temporary key
Usage: Encryption Key
Key is not exportable.
Key Data:
  307C300D 06092A86 4886F70D 01010105 00036B00 30680261 00AE125B 3DC8E36A
  8B74ED13 B2D2A788 D4DB2D07 F0A94B2C 85F79FBA 96EA1CAD 82FA83A2 5954F5E0
  A9D11280 C6007B28 9EA41364 919E7265 08CD7610 75E7FB1F D14F8484 206431F4
  F53393BB 155BEE64 94E6D448 385A97EB 900DA8FE ADF85D93 F9020301 0001

```

Figure 16 - R1's Public Key



- Using Wireshark, we can identify the HTTP message that requests the certificate in Figure 18 and the message that carries the requested certificate in Figure 19. We also extracted the certificate to analyse better its contents presented in Certificate General Info and Figure 21.

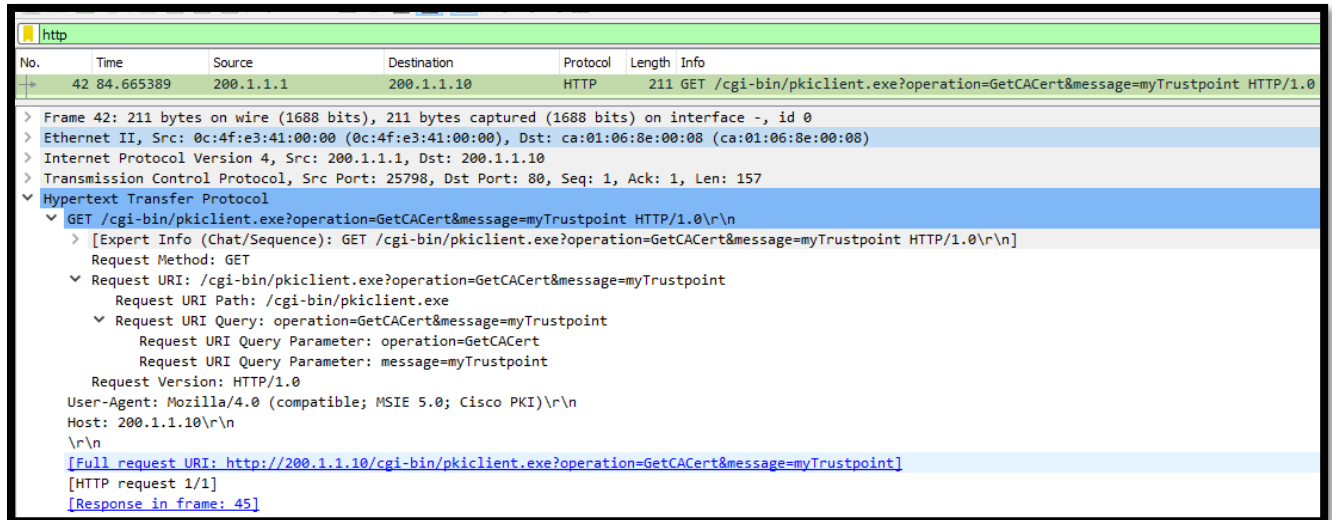


Figure 18 - GET certificate HTTP message

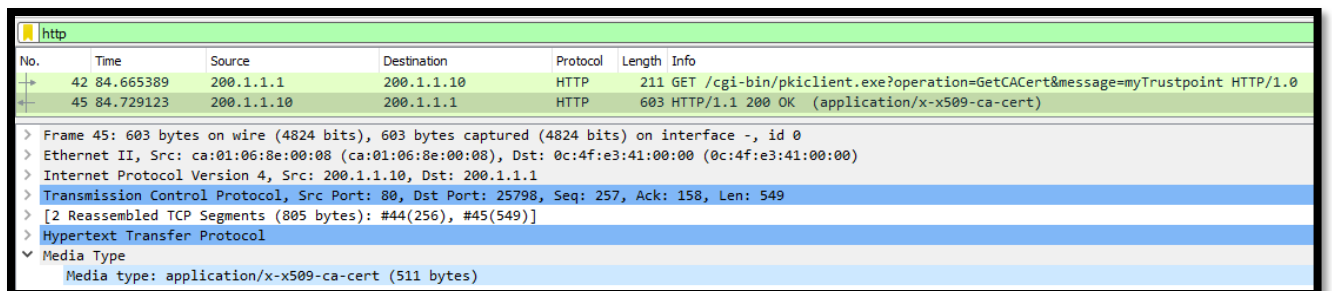


Figure 19 - HTTP OK message containing the CA certificate

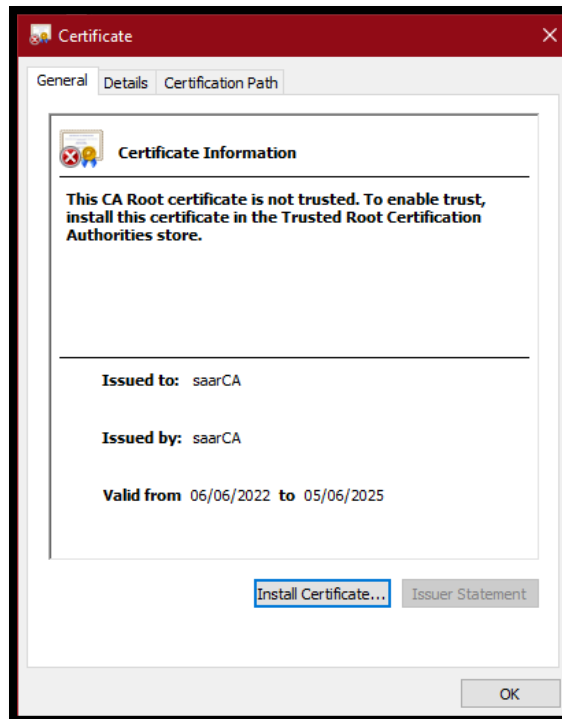


Figure 21 - Certificate General Info

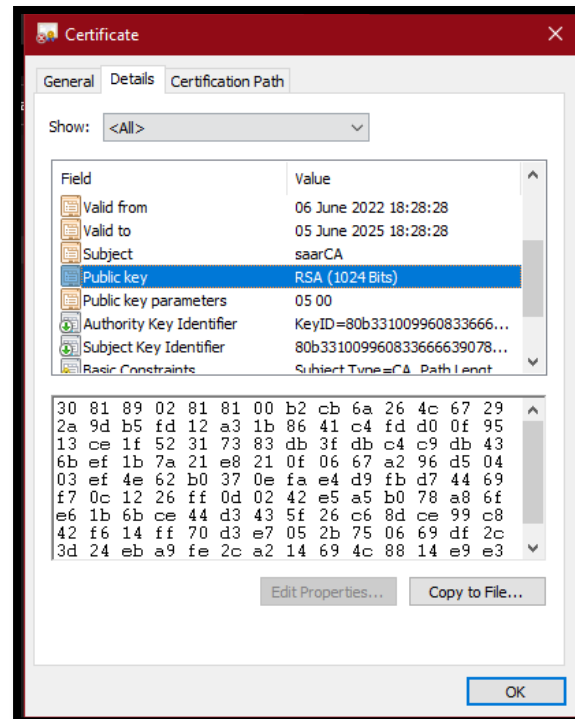


Figure 20 - Certificate Details

- Using an RSA signature authentication method, we now configure IPsec with Public Signature Keys using crypto maps at R1 and R2. This allows PC1 to communicate with PC2, as shown in Figure 22. In Figure 23, we can see that the main difference concerning IPsec with pre-shared keys is the authentication method type, which in this case is using RSA signatures in Digital certificates.

```
PC1> ping 192.168.2.100

192.168.2.100 icmp_seq=1 timeout
84 bytes from 192.168.2.100 icmp_seq=2 ttl=62 time=35.739 ms
84 bytes from 192.168.2.100 icmp_seq=3 ttl=62 time=21.390 ms
84 bytes from 192.168.2.100 icmp_seq=4 ttl=62 time=22.447 ms
84 bytes from 192.168.2.100 icmp_seq=5 ttl=62 time=21.566 ms
```

Figure 22 - Ping from PC1 to PC2

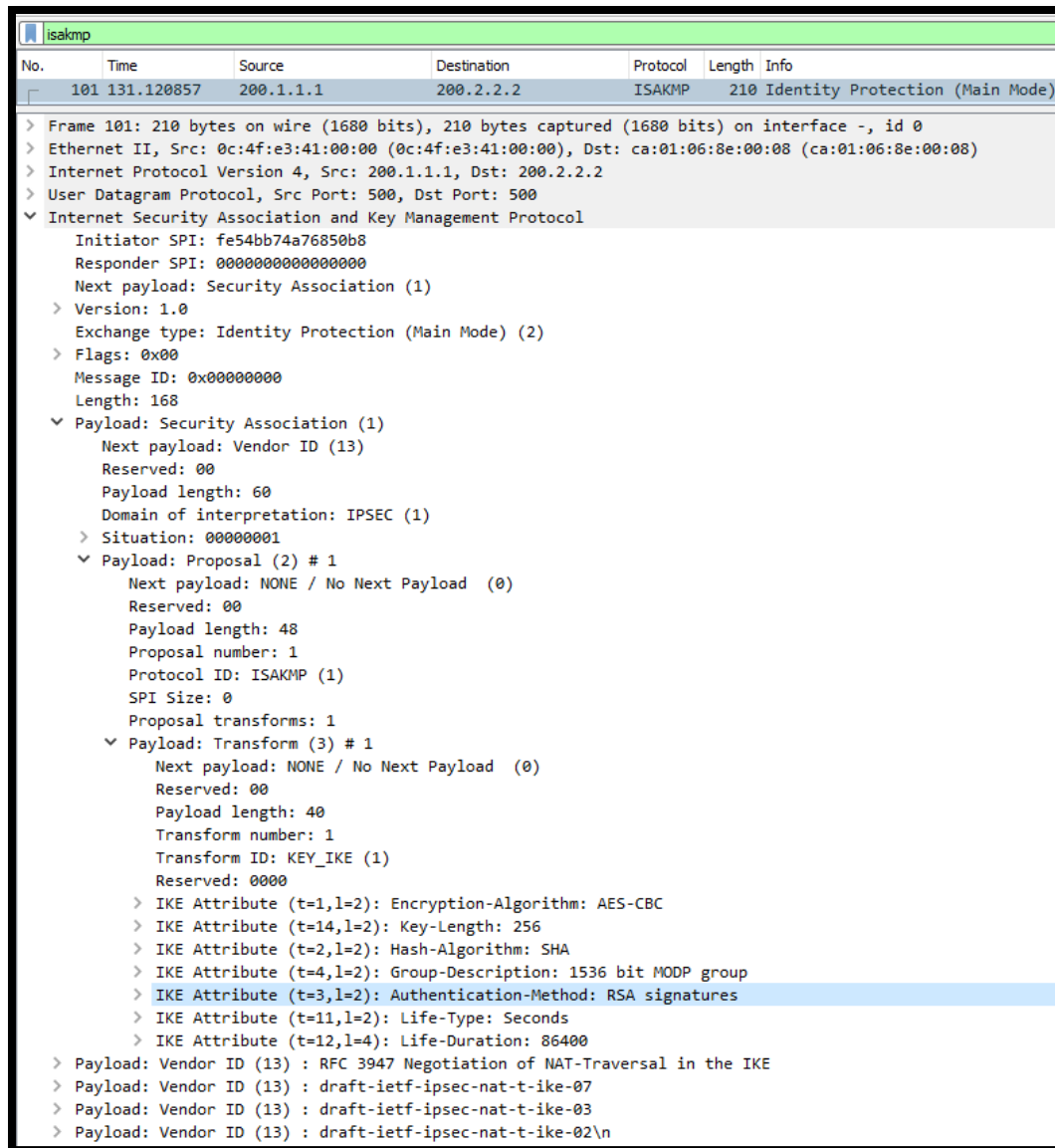


Figure 23 - IKE authentication method using Digital certificates

5. Now that the configurations are complete and we managed to communicate from PC1 to PC2, we can see that each router stores the public key of the other in Figure 24, which shows the public key of R2 at R1. We can also see in Figure 25 that ICMP messages exchanged between R1 and R2 cannot be identified since they are cyphered.

```

Router#show crypto key pubkey-chain rsa name Router
Key name: Router
Subject name(X.500 DN name): hostname=Router+serialNumber=93FVZFKVSS57NP31AMCWPP
Key id: 11
Serial number: 03
Usage: Signature Key
Source: Certificate
Data:
30819F30 0D06092A 864886F7 0D010101 05000381 8D003081 89028181 00A8CCAF
2BE215D7 19999D64 CBCD13A4 BD95FC72 5CC30544 8ECAC918 58433856 E8F0ACD7
8A546821 282A35D5 DFE74ED4 F43966FF 98ACC1A2 62EAA82C ACBC7C57 B2CA45B7
C3CDA18A 682DE2A6 860072AB 1E06C02D 929239AD 85A19A50 E6AC0BFD 4898228C
96CE484C FDA8F5DE 9A8ED796 03E0F950 AAA5ADD2 444AB00F CB0FF65F 59020301
0001

```

Figure 24 - R2 Public key at stored in R1

esp						
No.	Time	Source	Destination	Protocol	Length	Info
161	212.512533	200.1.1.1	200.2.2.2	ESP	166	ESP (SPI=0x1eb0f314)
162	212.550642	200.2.2.2	200.1.1.1	ESP	166	ESP (SPI=0x12153503)
164	213.575083	200.1.1.1	200.2.2.2	ESP	166	ESP (SPI=0x1eb0f314)
165	213.595133	200.2.2.2	200.1.1.1	ESP	166	ESP (SPI=0x12153503)
166	214.602250	200.1.1.1	200.2.2.2	ESP	166	ESP (SPI=0x1eb0f314)
167	214.616926	200.2.2.2	200.1.1.1	ESP	166	ESP (SPI=0x12153503)
> Frame 161: 166 bytes on wire (1328 bits), 166 bytes captured (1328 bits) on interface -, id 0 > Ethernet II, Src: 0c:4f:e3:41:00:00 (0c:4f:e3:41:00:00), Dst: ca:01:06:8e:00:08 (ca:01:06:8e:00:08) > Internet Protocol Version 4, Src: 200.1.1.1, Dst: 200.2.2.2 > Encapsulating Security Payload ESP SPI: 0x1eb0f314 (514913044) ESP Sequence: 1						

Figure 25 – Cyphered ICMP messages between R1 and R2

### 1.3. IPSec with NAT traversal

In this exercise, we test the IPSec tunnel with NAT Traversal. After the address configurations, we have configured **OSPF** as the routing protocol on the left side of the organisation and static routing between the two organisations. The complete configuration can be found in [Annex A](#).

#### Network Architecture:

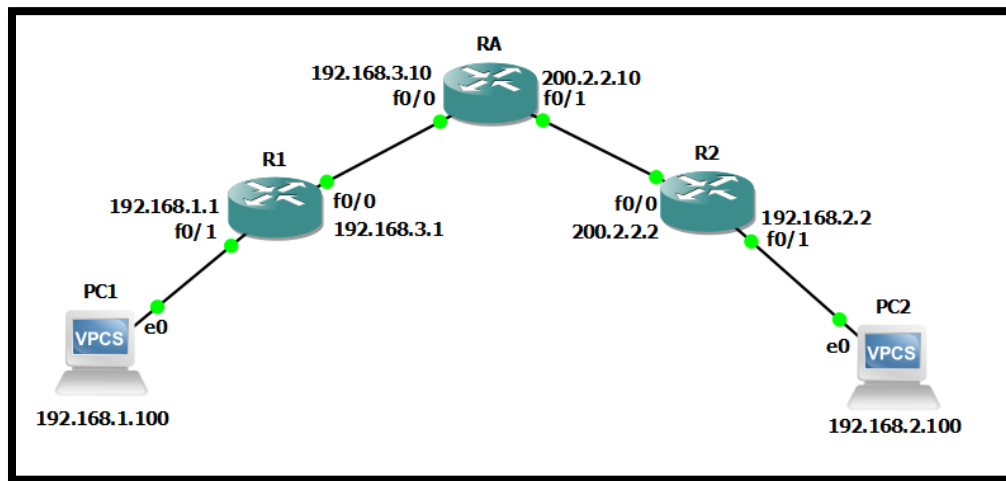


Figure 26 - IPSec with NAT traversal Network Architecture

In this architecture, the organisation's left side now includes two subnets: 192.168.1.0/24 and 192.168.3.0/24. The IPSec tunnel is still between R1 and R2, but RA is now the border router and includes PAT (i.e., NAT overload).

#### Proof-of-Concept:

Regarding the IPSec configuration, we followed the configuration provided for R2 and adapted it for R1, changing the pre-shared key IP and its peer. To better analyse the establishment of the IPSec tunnel, we configured a Wireshark capture between R1 and RA and between RA and R2.

1. We can see in Figure 27 the packet capture between R1 and RA (Private link). Here we can observe that the source is the IP **192.168.3.1** previously configured in the interface of R1. On the other hand, in Figure 28 (Public link), we can see that the source IP is **200.2.2.10**. Since RA has NAT configured, packets from the private network on the left will be translated to the public address **200.2.2.10**.

\*- [R1 FastEthernet0/0 to RA FastEthernet0/0]

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

isakmp

No.	Time	Source	Destination	Protocol	Length	Info
20	42.079096	192.168.3.1	200.2.2.2	ISAKMP	210	Identity Protection (Main Mode)
21	42.151562	200.2.2.2	192.168.3.1	ISAKMP	150	Identity Protection (Main Mode)
22	42.266215	192.168.3.1	200.2.2.2	ISAKMP	390	Identity Protection (Main Mode)
23	42.481525	200.2.2.2	192.168.3.1	ISAKMP	410	Identity Protection (Main Mode)
24	42.696551	192.168.3.1	200.2.2.2	ISAKMP	154	Identity Protection (Main Mode)
25	42.748053	200.2.2.2	192.168.3.1	ISAKMP	122	Identity Protection (Main Mode)
26	42.779004	192.168.3.1	200.2.2.2	ISAKMP	234	Quick Mode
27	42.830202	200.2.2.2	192.168.3.1	ISAKMP	234	Quick Mode
28	42.860794	192.168.3.1	200.2.2.2	ISAKMP	106	Quick Mode

<

> Frame 20: 210 bytes on wire (1680 bits), 210 bytes captured (1680 bits) on interface -, id 0  
 > Ethernet II, Src: ca:01:09:68:00:08 (ca:01:09:68:00:08), Dst: c2:03:09:94:00:00 (c2:03:09:94:00:00)  
 > Internet Protocol Version 4, Src: 192.168.3.1, Dst: 200.2.2.2  
 > User Datagram Protocol, Src Port: 500, Dst Port: 500  
 > Internet Security Association and Key Management Protocol

Figure 27 - Packet capture between R1 and RA (Private link)

\*- [RA FastEthernet0/1 to R2 FastEthernet0/0]

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

isakmp

No.	Time	Source	Destination	Protocol	Length	Info
10	36.576535	200.2.2.10	200.2.2.2	ISAKMP	210	Identity Protection (Main Mode)
11	36.618256	200.2.2.2	200.2.2.10	ISAKMP	150	Identity Protection (Main Mode)
12	36.753291	200.2.2.10	200.2.2.2	ISAKMP	390	Identity Protection (Main Mode)
13	36.948010	200.2.2.2	200.2.2.10	ISAKMP	410	Identity Protection (Main Mode)
14	37.183683	200.2.2.10	200.2.2.2	ISAKMP	154	Identity Protection (Main Mode)
15	37.214551	200.2.2.2	200.2.2.10	ISAKMP	122	Identity Protection (Main Mode)
16	37.266072	200.2.2.10	200.2.2.2	ISAKMP	234	Quick Mode
17	37.296283	200.2.2.2	200.2.2.10	ISAKMP	234	Quick Mode
18	37.347974	200.2.2.10	200.2.2.2	ISAKMP	106	Quick Mode

<

> Frame 10: 210 bytes on wire (1680 bits), 210 bytes captured (1680 bits) on interface -, id 0  
 > Ethernet II, Src: c2:03:09:94:00:01 (c2:03:09:94:00:01), Dst: ca:02:09:78:00:08 (ca:02:09:78:00:08)  
 > Internet Protocol Version 4, Src: 200.2.2.10, Dst: 200.2.2.2  
 > User Datagram Protocol, Src Port: 500, Dst Port: 500  
 > Internet Security Association and Key Management Protocol

Figure 28 - Packet capture on the public link

- We have the third and fourth IKE messages in Figure 29 and Figure 30, respectively. We can observe that both packets contain two **NAT-D Payloads**. Furthermore, we can see that these payloads have a Hash value. The first uses the source IP and port for the hash, and the second one uses the destination IP and source. The peer will use these values to verify if NAT was used. If the hash values match, then NAT was not used; if the hashes do not match, then NAT was indeed employed.

In this situation, the second payload will not match the one calculated by the other end, as the packet received by this peer has a different source IP than the one previously used for the hash. This way, the peer will notice that NAT is being used.

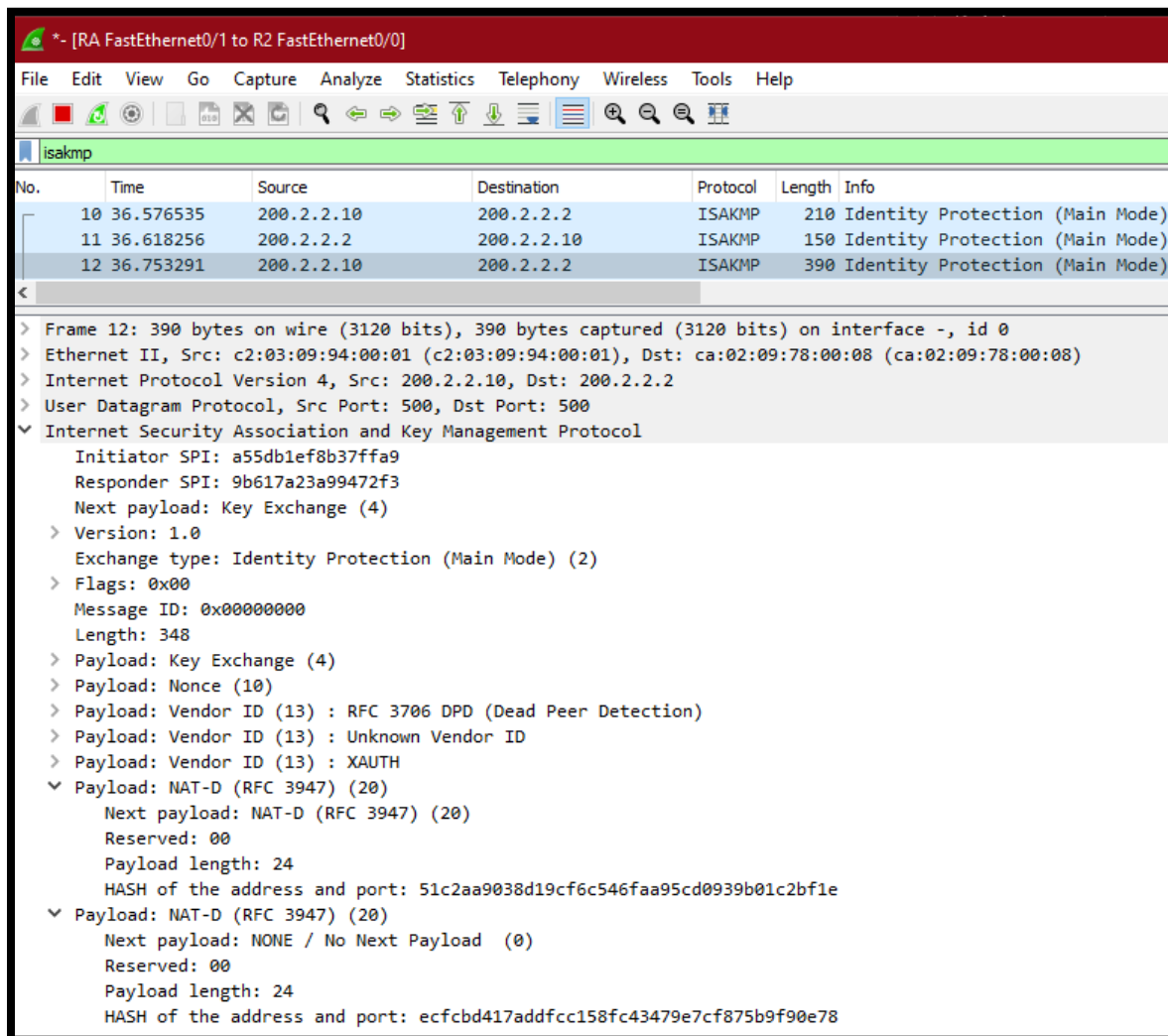


Figure 29 - Third ISAKMP message with NAT-D Payload

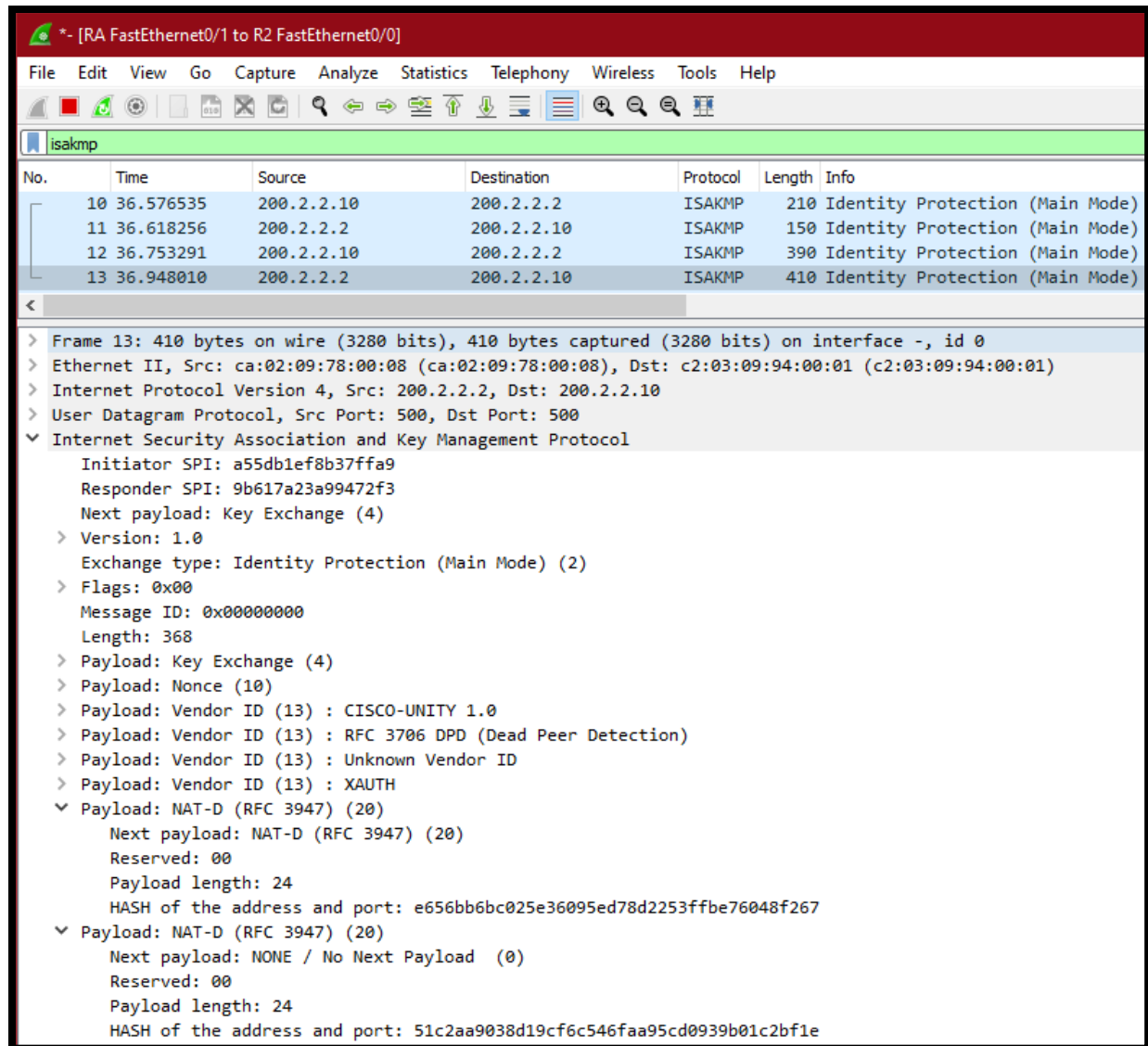


Figure 30 - Fourth ISAKMP message with NAT-D Payload

- Since **NAT** is detected, **UDP port 4500** will be used from now on, starting in the fifth packet, as shown in Figure 31.



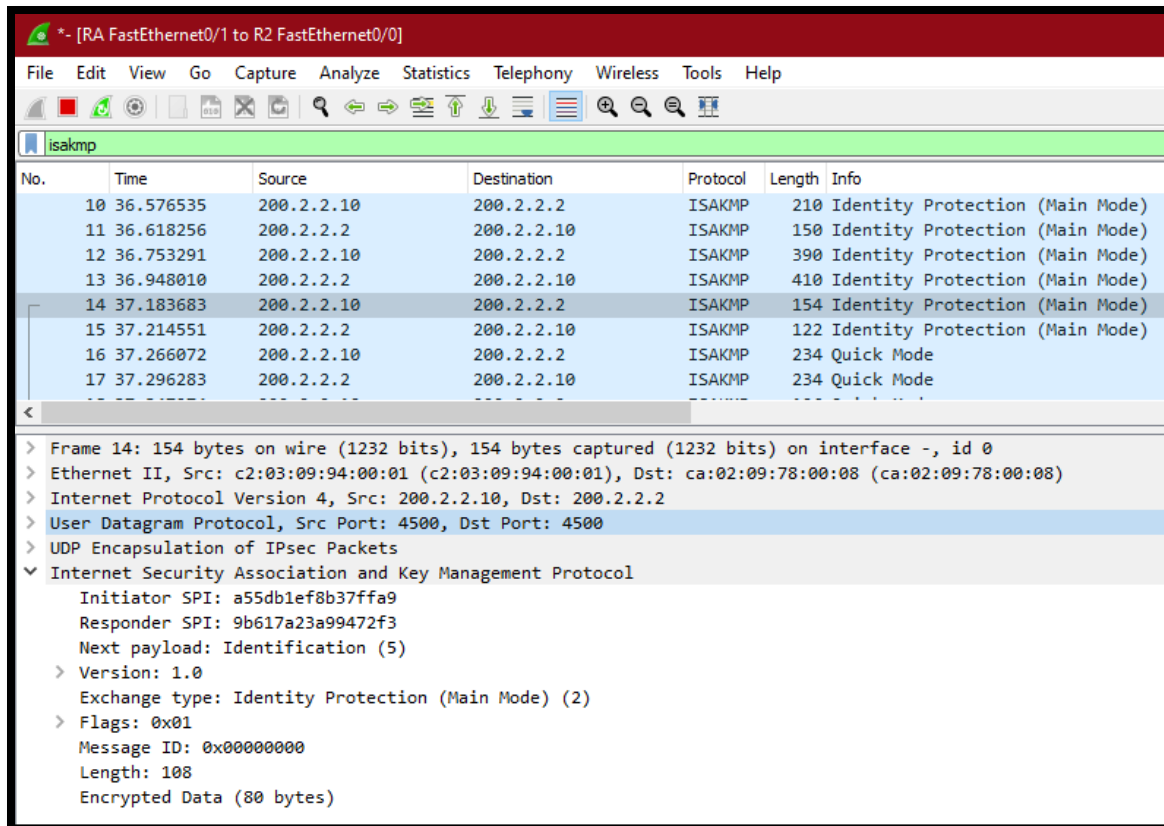


Figure 31 - Fifth ISAKMP packet with port 4500

- In Figure 32 and Figure 33, we can see the ESP packets in the private and public networks, respectively. As noticed earlier, it is possible to observe that the communication in both networks uses UDP port 4500, allowing the packets to transverse the NAT.

\*- [R1 FastEthernet0/0 to RA FastEthernet0/0]

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

esp

No.	Time	Source	Destination	Protocol	Length	Info
29	43.838705	192.168.3.1	200.2.2.2	ESP	174	ESP (SPI=0x0240a930)
30	43.889956	200.2.2.2	192.168.3.1	ESP	174	ESP (SPI=0xaea13973)
32	44.916882	192.168.3.1	200.2.2.2	ESP	174	ESP (SPI=0x0240a930)
33	44.947123	200.2.2.2	192.168.3.1	ESP	174	ESP (SPI=0xaea13973)
35	45.953181	192.168.3.1	200.2.2.2	ESP	174	ESP (SPI=0x0240a930)
36	45.983528	200.2.2.2	192.168.3.1	ESP	174	ESP (SPI=0xaea13973)
38	46.998385	192.168.3.1	200.2.2.2	ESP	174	ESP (SPI=0x0240a930)
39	47.029194	200.2.2.2	192.168.3.1	ESP	174	ESP (SPI=0xaea13973)

<

> Frame 29: 174 bytes on wire (1392 bits), 174 bytes captured (1392 bits) on interface -, id 0  
 > Ethernet II, Src: ca:01:09:68:00:08 (ca:01:09:68:00:08), Dst: c2:03:09:94:00:00 (c2:03:09:94:00:00)  
 > Internet Protocol Version 4, Src: 192.168.3.1, Dst: 200.2.2.2  
 > User Datagram Protocol, Src Port: 4500, Dst Port: 4500  
   Source Port: 4500  
   Destination Port: 4500  
   Length: 140  
   [Checksum: [missing]]  
   [Checksum Status: Not present]  
   [Stream index: 1]  
   > [Timestamps]  
     UDP payload (132 bytes)  
   UDP Encapsulation of IPsec Packets  
   > Encapsulating Security Payload

Figure 32 - ESP packets in the private network

\*- [RA FastEthernet0/1 to R2 FastEthernet0/0]

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

esp

No.	Time	Source	Destination	Protocol	Length	Info
19	38.325756	200.2.2.10	200.2.2.2	ESP	174	ESP (SPI=0x0240a930)
20	38.361997	200.2.2.2	200.2.2.10	ESP	174	ESP (SPI=0xaea13973)
22	39.403449	200.2.2.10	200.2.2.2	ESP	174	ESP (SPI=0x0240a930)
23	39.419805	200.2.2.2	200.2.2.10	ESP	174	ESP (SPI=0xaea13973)
25	40.439746	200.2.2.10	200.2.2.2	ESP	174	ESP (SPI=0x0240a930)
26	40.454503	200.2.2.2	200.2.2.10	ESP	174	ESP (SPI=0xaea13973)
27	41.485457	200.2.2.10	200.2.2.2	ESP	174	ESP (SPI=0x0240a930)
28	41.499583	200.2.2.2	200.2.2.10	ESP	174	ESP (SPI=0xaea13973)

<

> Frame 19: 174 bytes on wire (1392 bits), 174 bytes captured (1392 bits) on interface -, id 0  
 > Ethernet II, Src: c2:03:09:94:00:01 (c2:03:09:94:00:01), Dst: ca:02:09:78:00:08 (ca:02:09:78:00:08)  
 > Internet Protocol Version 4, Src: 200.2.2.10, Dst: 200.2.2.2  
 > User Datagram Protocol, Src Port: 4500, Dst Port: 4500  
   Source Port: 4500  
   Destination Port: 4500  
   Length: 140  
   [Checksum: [missing]]  
   [Checksum Status: Not present]  
   [Stream index: 1]  
   > [Timestamps]  
     UDP payload (132 bytes)  
   UDP Encapsulation of IPsec Packets  
   > Encapsulating Security Payload

Figure 33 - ESP packets in the public network

## 1.4. GRE over IPSec

Generic Routing Encapsulation protocol is a simple IP packet encapsulation protocol. It works by encapsulating a payload inside an outer IP packet. Although helpful, GRE is not secure, and IPSec only supports unicast traffic. The solution for having a secure tunnel capable of transporting unicast, multicast, and broadcast IP packets is GRE over IPSec, where broadcast and multicast traffic is encapsulated inside a unicast packet processed by IPSec.

### Network Architecture:

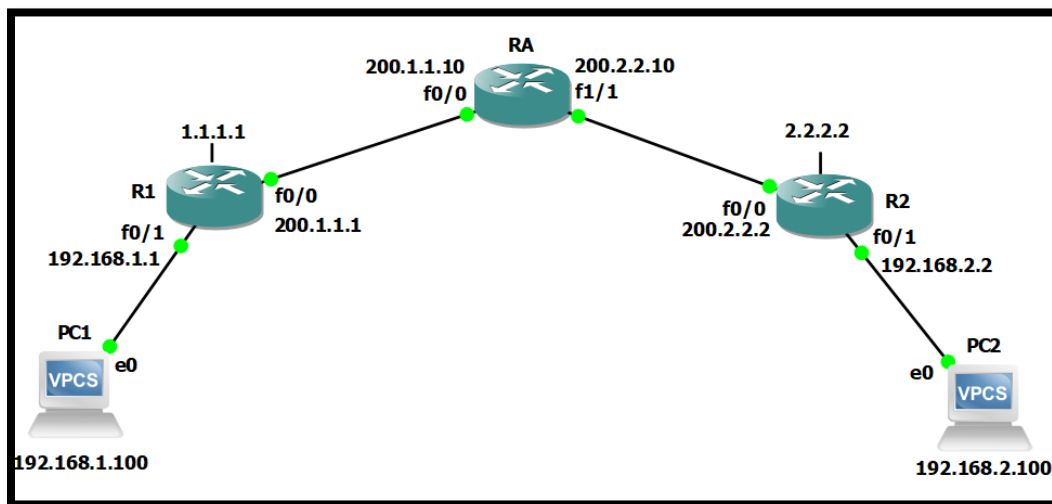


Figure 34 - GRE over IPSec Network Architecture

In this exercise, we configure the network of Figure 34 to support GRE over IPSec (AH protocol in tunnel mode,) where the organisation on the network uses OSPF for routing. The secure tunnel is between R1 and R2, and the public network also uses OSPF. The complete configuration can be found in [Annex A](#).

### Proof-of-Concept:

1. After configuring the initial network structures, including the two OSPF areas, we configured the security profile using GRE over IPSec with AH tunnel mode. We captured an ICMP packet at the public network when doing a ping from PC1 to PC2. In this packet, presented in Figure 35, we can identify three encapsulation layers.

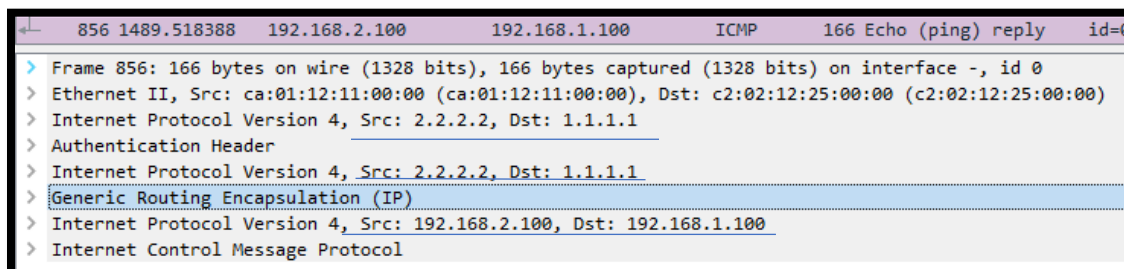


Figure 35 - ICMP Packet from PC1 to PC2 (AH Tunnel mode)

- The most in-depth layer holds the ICMP packet, which is then encapsulated inside a GRE packet identified by a new IP GRE header, which is again encapsulated and given another IP header provided by the IPsec's tunnel mode. The diagram below provides a visual representation of this explanation.

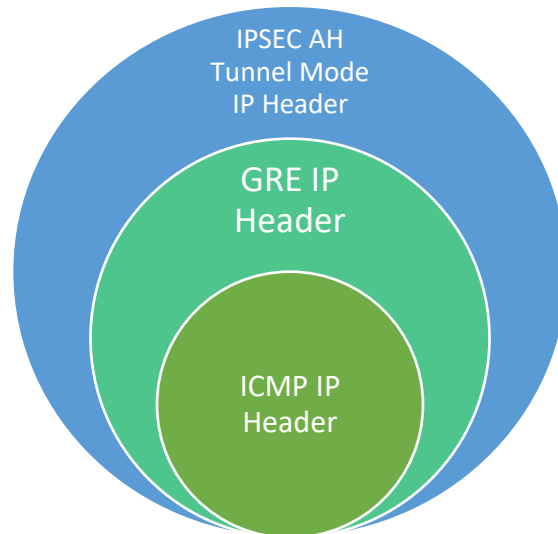


Figure 36 - GRE over IPsec AH Tunnel Mode packet structure

- The same encapsulation type is also observed in the OSPF packets sent through the tunnel. Figure 38 shows two types of OSPF Hello Packets: one type from the private network and another from the public network. The main difference between them is that the Hello Packets from the public network, presented in Figure 37, only have one IP header since they are not transmitted through GRE over IPsec.

ospf						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	200.1.1.10	224.0.0.5	OSPF	94	Hello Packet
> Frame 1: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface -, id 0 > Ethernet II, Src: ca:01:12:11:00:00 (ca:01:12:11:00:00), Dst: IPv4mcast_05 (01:00:5e:00:00:05) > Internet Protocol Version 4, Src: 200.1.1.10, Dst: 224.0.0.5 > Open Shortest Path First						

Figure 37 - OSPF Hello Packet Public network

ospf						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	200.1.1.10	224.0.0.5	OSPF	94	Hello Packet
2	0.131697	192.168.2.2	224.0.0.5	OSPF	162	Hello Packet
3	0.153498	192.168.1.1	224.0.0.5	OSPF	162	Hello Packet
4	0.698055	200.1.1.1	224.0.0.5	OSPF	94	Hello Packet
8	9.114678	200.1.1.10	224.0.0.5	OSPF	94	Hello Packet
9	10.141926	192.168.2.2	224.0.0.5	OSPF	162	Hello Packet
10	10.152899	192.168.1.1	224.0.0.5	OSPF	162	Hello Packet
11	10.700356	200.1.1.1	224.0.0.5	OSPF	94	Hello Packet
14	18.440631	200.1.1.10	224.0.0.5	OSPF	94	Hello Packet
15	20.144121	192.168.2.2	224.0.0.5	OSPF	162	Hello Packet
16	20.153049	192.168.1.1	224.0.0.5	OSPF	162	Hello Packet
17	20.695609	200.1.1.1	224.0.0.5	OSPF	94	Hello Packet
21	28.347811	200.1.1.10	224.0.0.5	OSPF	94	Hello Packet
22	30.130414	192.168.2.2	224.0.0.5	OSPF	162	Hello Packet
23	30.149225	192.168.1.1	224.0.0.5	OSPF	162	Hello Packet
24	30.701741	200.1.1.1	224.0.0.5	OSPF	94	Hello Packet
27	37.443289	200.1.1.10	224.0.0.5	OSPF	94	Hello Packet
28	40.140106	192.168.2.2	224.0.0.5	OSPF	162	Hello Packet
29	40.150487	192.168.1.1	224.0.0.5	OSPF	162	Hello Packet
30	40.691599	200.1.1.1	224.0.0.5	OSPF	94	Hello Packet
34	47.179411	200.1.1.10	224.0.0.5	OSPF	94	Hello Packet

>	Frame 9: 162 bytes on wire (1296 bits), 162 bytes captured (1296 bits) on interface -, id 0
>	Ethernet II, Src: ca:01:12:11:00:00 (ca:01:12:11:00:00), Dst: c2:02:12:25:00:00 (c2:02:12:25:00:00)
>	Internet Protocol Version 4, Src: 2.2.2.2, Dst: 1.1.1.1
>	Authentication Header
>	Internet Protocol Version 4, Src: 2.2.2.2, Dst: 1.1.1.1
>	Generic Routing Encapsulation (IP)
>	Internet Protocol Version 4, Src: 192.168.2.2, Dst: 224.0.0.5
>	Open Shortest Path First

Figure 38 - OSPF Hello Packet Private network

- By analysing the OSPF LSDB related to the internal network presented in Figure 39, we can also conclude that two types of LSAs are used. Each router generates the Router LSA for each area it is located. In the link-state ID, we can find the originating router's ID. The second type are Network LSAs that are generated by the DR, where the link-state ID is the interface IP address of the DR.

```
OSPF Router with ID (200.2.2.2) (Process ID 2)

Router Link States (Area 0)

LS age: 263
Options: (No TOS-capability, DC)
LS Type: Router Links
Link State ID: 200.1.1.1
Advertising Router: 200.1.1.1
LS Seq Number: 80000005
Checksum: 0x3D6D
Length: 48
Number of Links: 2

  Link connected to: a Stub Network
    (Link ID) Network/subnet number: 192.168.1.0
    (Link Data) Network Mask: 255.255.255.0
      Number of TOS metrics: 0
      TOS 0 Metrics: 10

  Link connected to: another Router (point-to-point)
    (Link ID) Neighboring Router ID: 200.2.2.2
    (Link Data) Router Interface address: 0.0.0.9
      Number of TOS metrics: 0
      TOS 0 Metrics: 11111

LS age: 244
Options: (No TOS-capability, DC)
LS Type: Router Links
Link State ID: 200.2.2.2
Advertising Router: 200.2.2.2
LS Seq Number: 80000005
Checksum: 0xD0D5
Length: 48
Number of Links: 2

  Link connected to: a Stub Network
    (Link ID) Network/subnet number: 192.168.2.0
    (Link Data) Network Mask: 255.255.255.0
      Number of TOS metrics: 0
      TOS 0 Metrics: 10

  Link connected to: another Router (point-to-point)
    (Link ID) Neighboring Router ID: 200.1.1.1
    (Link Data) Router Interface address: 0.0.0.9
      Number of TOS metrics: 0
      TOS 0 Metrics: 11111
```

Figure 39 - Router 2 OSPF LSDB Private network

5. Now, we changed the IPSec protocol to ESP and looked at the packets sent while pingging PC2 from PC1. As we can see in Figure 40, it is impossible to distinguish the ICMP packets from the OSPF packets of the private network, as all these packets are cyphered from the point of view of the public network.

No.	Time	Source	Destination	Protocol	Length	Info
103	118.896983	200.1.1.10	224.0.0.5	OSPF	94	Hello Packet
104	119.337242	1.1.1.1	2.2.2.2	ESP	182	ESP (SPI=0x3e856cc2)
105	120.002877	200.1.1.1	224.0.0.5	OSPF	94	Hello Packet
106	122.677002	2.2.2.2	1.1.1.1	ESP	182	ESP (SPI=0x00d5fd8c)
107	123.404005	c2:02:12:25:00:00	c2:02:12:25:00:00	LOOP	60	Reply
108	124.182350	1.1.1.1	2.2.2.2	ESP	182	ESP (SPI=0x3e856cc2)
109	126.179626	1.1.1.1	2.2.2.2	ESP	182	ESP (SPI=0x3e856cc2)
110	126.220614	2.2.2.2	1.1.1.1	ESP	182	ESP (SPI=0x00d5fd8c)
111	127.235296	1.1.1.1	2.2.2.2	ESP	182	ESP (SPI=0x3e856cc2)
112	127.276411	2.2.2.2	1.1.1.1	ESP	182	ESP (SPI=0x00d5fd8c)
113	128.107806	200.1.1.10	224.0.0.5	OSPF	94	Hello Packet
114	128.107831	ca:01:12:11:00:00	ca:01:12:11:00:00	LOOP	60	Reply
115	128.292005	1.1.1.1	2.2.2.2	ESP	182	ESP (SPI=0x3e856cc2)
116	128.333090	2.2.2.2	1.1.1.1	ESP	182	ESP (SPI=0x00d5fd8c)
117	129.343224	1.1.1.1	2.2.2.2	ESP	182	ESP (SPI=0x3e856cc2)
118	129.353752	1.1.1.1	2.2.2.2	ESP	182	ESP (SPI=0x3e856cc2)
119	129.395587	2.2.2.2	1.1.1.1	ESP	182	ESP (SPI=0x00d5fd8c)
120	129.995485	200.1.1.1	224.0.0.5	OSPF	94	Hello Packet
121	132.675189	2.2.2.2	1.1.1.1	ESP	182	ESP (SPI=0x00d5fd8c)
122	133.399214	c2:02:12:25:00:00	c2:02:12:25:00:00	LOOP	60	Reply

Figure 40 - R1 to RA connection using ESP tunnel mode

6. We now change the IPSec protocol back to AH and the IPSec mode to transport to see the differences between transport and tunnel mode. By analysing the ICMP and OSPF packets, we can see several changes regarding encapsulation. In the ICMP and OSPF packets from the private networks, presented in Figure 42 and Figure 43, there are only two IP headers, concluding that the repeated AH IP header has been removed. We can see how this mode increases the efficiency of the network by removing the repeated header and therefore conclude that using transport mode can be more appropriate than tunnel mode when using GRE over IPSec.

87	42.151012	200.1.1.10	224.0.0.5	OSPF	94	Hello Packet
<						
> Frame 87: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface -, id 0						
> Ethernet II, Src: ca:01:12:11:00:00 (ca:01:12:11:00:00), Dst: IPv4mcast_05 (01:00:5e:00:00:05)						
> Internet Protocol Version 4, Src: 200.1.1.10, Dst: 224.0.0.5						
> Open Shortest Path First						

Figure 41 - OSPF Hello Public Network transport mode

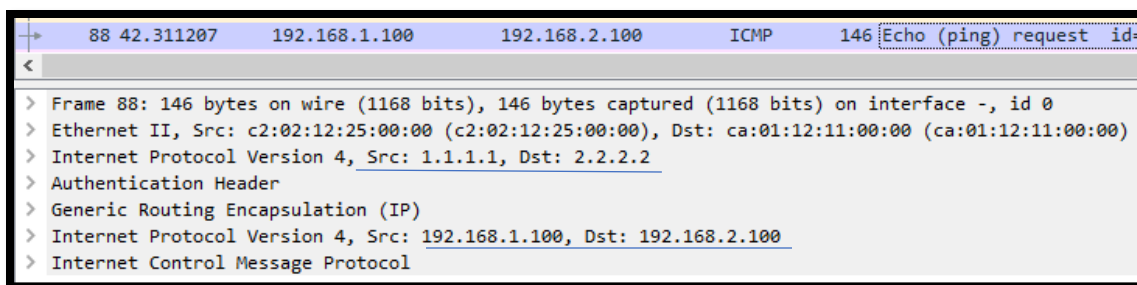


Figure 42 - ICMP packet PC1 to PC2 AH transport mode

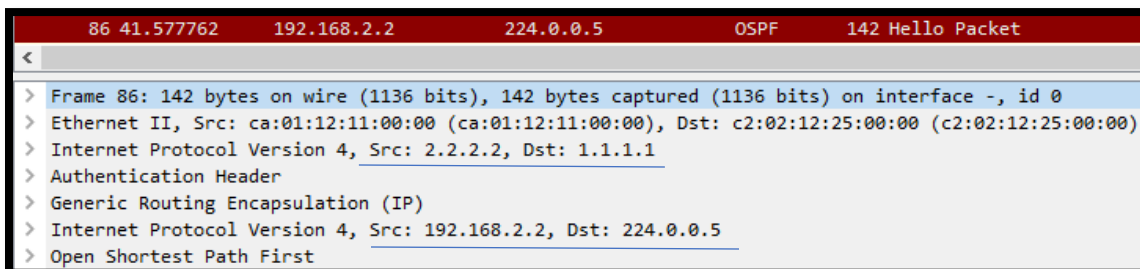


Figure 43 - OSPF packet Private Network AH transport mode



### 1.5. DMVPN Phase 3

Dynamic Multipoint Virtual Point Network presents a more scalable and dynamic VPN with Hub and Spoke topology. It allows the configuration of a single GRE tunnel interface and one IPsec profile on the hub router that manages all other routers. This feature automatically creates tunnels from hub to spoke or spoke to spoke. It resorts to NHRP, and public addresses are also called NBMA (Public addresses are also called NBMA (Non-Broadcast Multiple Access)). (Network Lessons, 2022)

This technology has three different phases corresponding to three configuration options with increasing levels of flexibility. In this lab, we will present the third and most flexible phase.

#### Network Architecture:

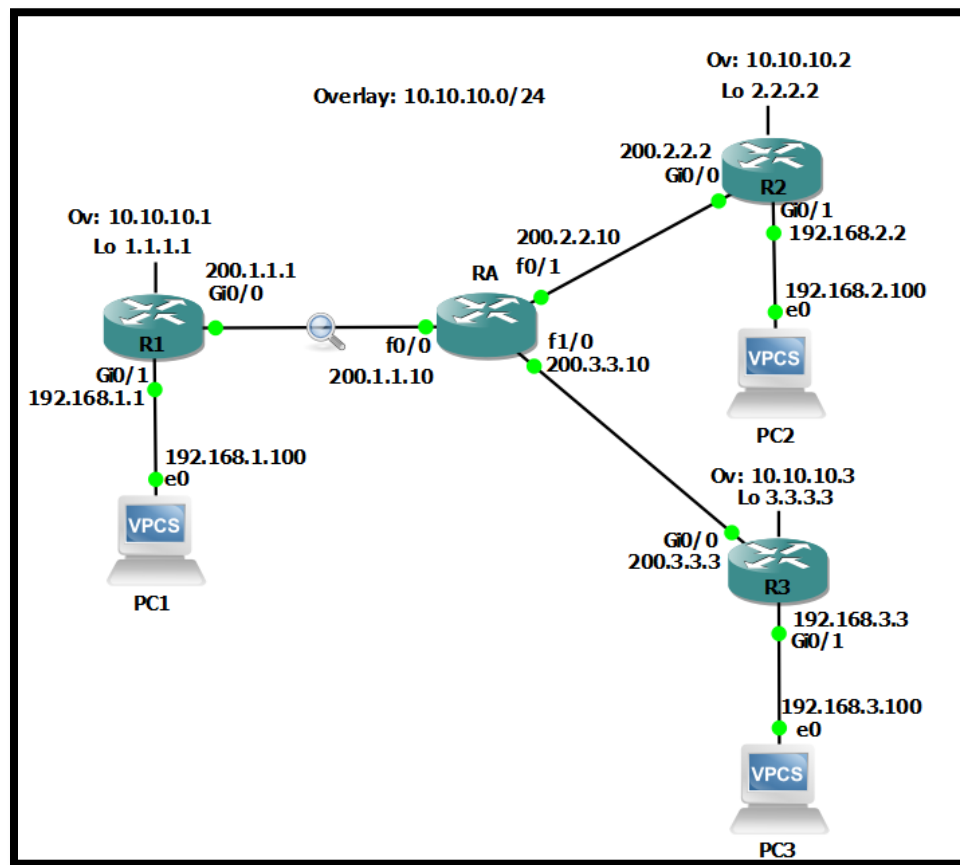


Figure 44 - DMVPN Network Architecture

This architecture represents an organisation's network with three sites: PC1, PC2 and PC3. In this DMVPN lab, R1 is the Hub, and R2 and R3 both are Spoke routers; all these three routers are Cisco IOSv routers. The complete configuration of this lab can be found in [Annex A](#).

**Proof-of-Concept:**

After implementing the required IP addresses and basic network configurations, we configured the DMVPN Phase 3 solution with NHRP redirects and shortcuts. In this lab exercise, the routing protocols at the organisation network are RIPv2 and OSPF in the outside network.

1. When both OSPF and RIPv2 are implemented correctly in the network, we start by configuring the Hub's (R1) tunnel interface with an NHRP multicast dynamic map with redirect. Then we configure both Spokes' tunnels (R2 and R3) with an NHRP multicast map and shortcut to the Hub.
2. By analysing the RIP and OSPF packets presented in Figure 45 and Figure 46, we could see that their outer IP addresses are different in the public network. RIP packets contain an external IP of the overlay network built on R1 as the source IP address and employ a RIPv2 multicast address (224.0.0.9) as the destination IP, sending routing information to all RIPv2 routers on the private network. Where OSPF uses R1's public interface IP address as the source address and an OSPF multicast address (224.0.0.5) as a destination address, sending Hello packets to all OSPF routers on the public network.

18 29.323981	10.10.10.1	224.0.0.9	RIPv2	90 Response
> Frame 18: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface -, id 0 > Ethernet II, Src: 0c:be:9e:79:00:00 (0c:be:9e:79:00:00), Dst: c2:01:5d:29:00:00 (c2:01:5d:29:00:00) > Internet Protocol Version 4, Src: 1.1.1.1, Dst: 2.2.2.2 > Generic Routing Encapsulation (IP) > Internet Protocol Version 4, Src: 10.10.10.1, Dst: 224.0.0.9 > User Datagram Protocol, Src Port: 520, Dst Port: 520 > Routing Information Protocol				
Command: Response (2) Version: RIPv2 (2) > IP Address: 0.0.0.0, Metric: 1 Address Family: IP (2) Route Tag: 0 IP Address: 0.0.0.0 Netmask: 0.0.0.0 Next Hop: 0.0.0.0 Metric: 1				

Figure 45 - RIPv2 packet on the public network

20 37.113562	200.1.1.1	224.0.0.5	OSPF	94 Hello Packet
> Frame 20: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface -, id 0 > Ethernet II, Src: 0c:be:9e:79:00:00 (0c:be:9e:79:00:00), Dst: IPv4mcast_05 (01:00:5e:00:00:05) > Internet Protocol Version 4, Src: 200.1.1.1, Dst: 224.0.0.5 > Open Shortest Path First				

Figure 46 - OSPF packet on the public network

- To better understand its functionality, we removed the command **ip summary-address rip 0.0.0.0 0.0.0.0** from the tunnel configuration of R1. This command summarises routes in RIPv2 to improve scalability and efficiency. Using this option means there is no entry for child routes in the RIP routing table, reducing the table size and allowing the router to handle more paths. Figure 47 and Figure 48 show a RIP packet being sent without the ip summary-address option and R1's routing table, respectively.

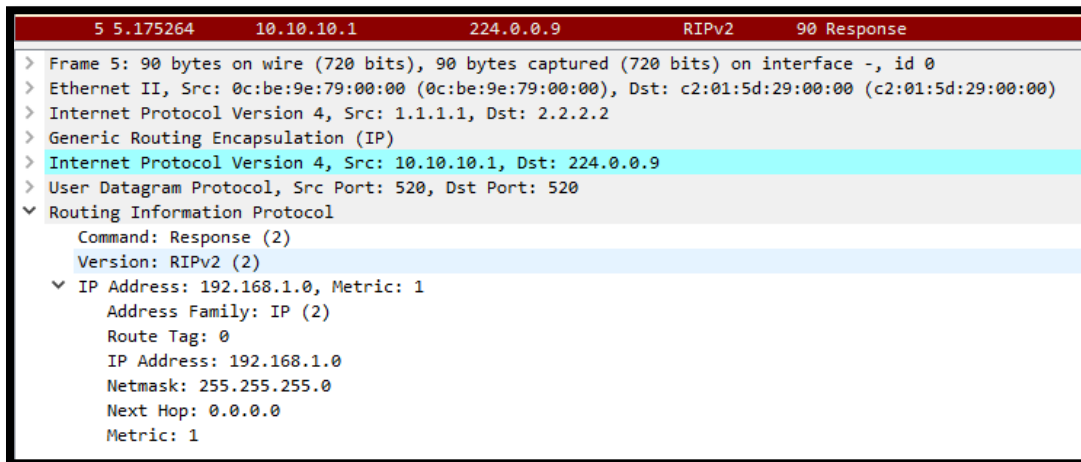


Figure 47 - RIPv2 packet with no ip summary-address

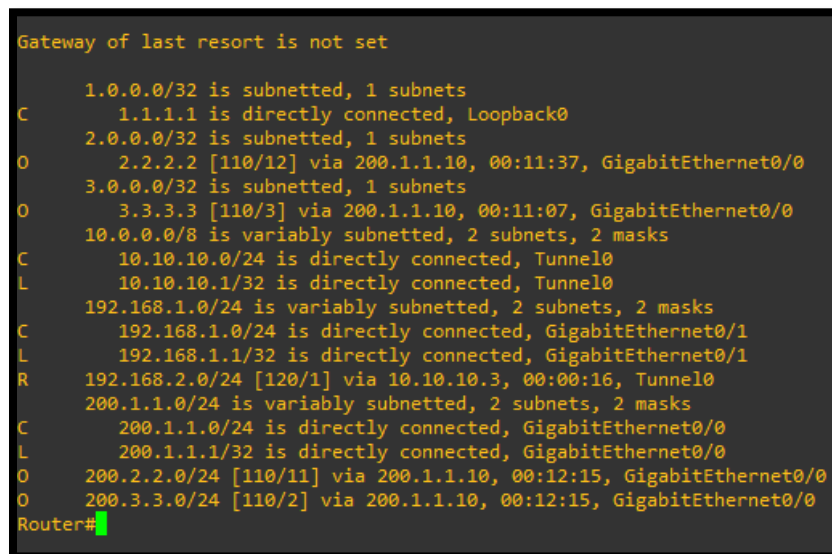


Figure 48 - R1's Routing table

- In addition to such a configuration, we ran the command **no ip split-horizon**. Split-horizon prohibits a router from advertising a route through an interface that the router uses to reach the destination. This is a convenient feature to prevent loops in a network. In Figure 49, we can see the Hub (R1) sending a RIPv2 packet announcing all the routes it has learned back into the network, which would never occur with split-horizon. (fryadmin, 2011)

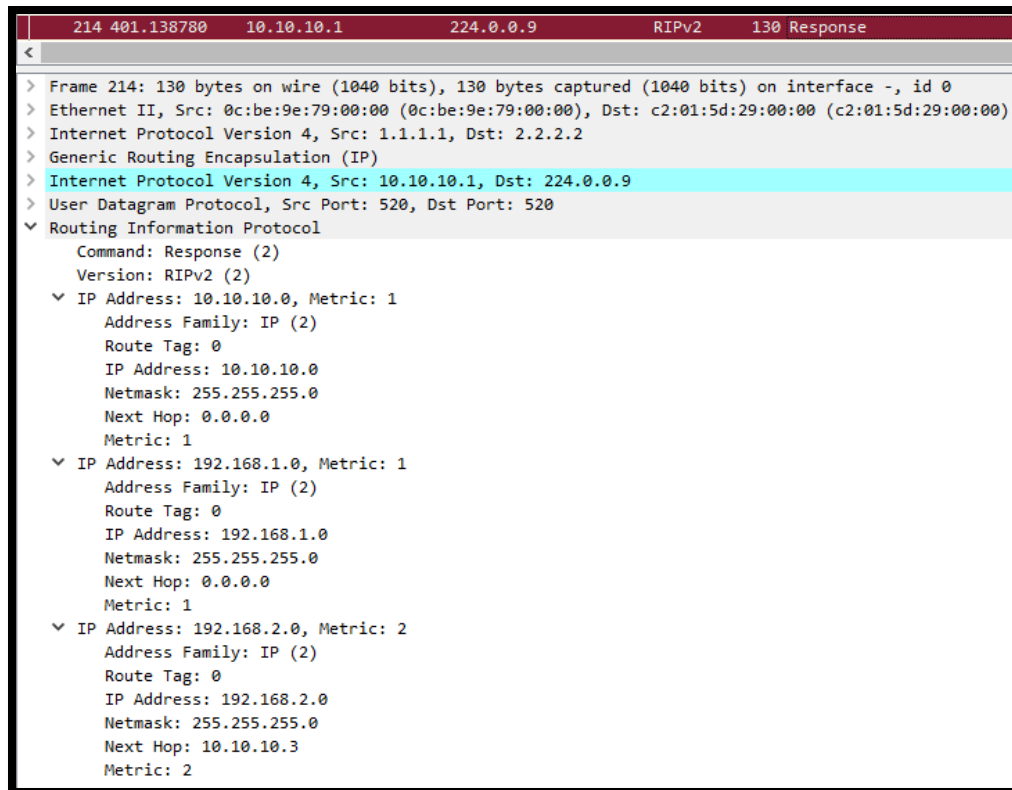


Figure 49 – RIPv2 packet with no Split-horizon and no summary-address

- While analysing the traffic of the DMVPN, we could also capture packets such as the one in Figure 50. NHRP or Next Hop Resolution Protocol allows a Next Hop Client (NHC) to dynamically register with Next Hop Servers. In this case, the NHC are the spoke routers, and the NHS is the hub router. This protocol uses an NHRP cache that stores static and dynamic entries with mapping information from packets such as the one in Figure 50. The NHRP cache of Router 1 can be seen in Figure 51. (Trenner, 2021)

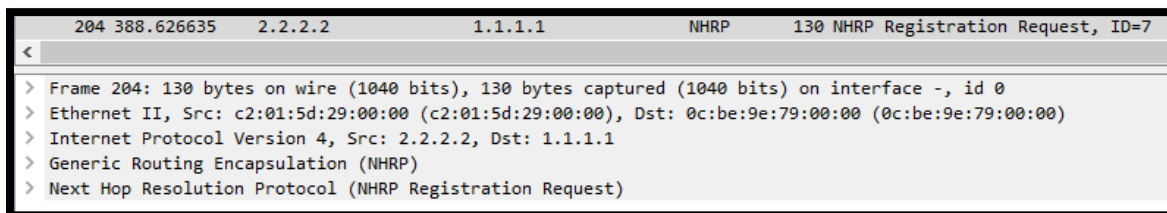


Figure 50 - NHRP registration request packet

```
Router#show dmvpn
Legend: Attrb --> S - Static, D - Dynamic, I - Incomplete
        N - NATed, L - Local, X - No Socket
        T1 - Route Installed, T2 - Nexthop-override
        C - CTS Capable, I2 - Temporary
        # Ent --> Number of NHRP entries with same NBMA peer
        NHS Status: E --> Expecting Replies, R --> Responding, W --> Waiting
        UpDn Time --> Up or Down Time for a Tunnel
=====

Interface: Tunnel0, IPv4 NHRP Details
Type:Hub, NHRP Peers:2,

# Ent  Peer NBMA Addr Peer Tunnel Add State  UpDn Tm Attrb
-----
   1 2.2.2.2          10.10.10.2  UP 00:01:15  D
   1 3.3.3.3          10.10.10.3  UP 00:01:03  D
```

Figure 51 - R1's NHRP cache

## 1.6. DMVPN over IPSec

Since DMVPN is usually used with the internet as the underlay network, it is best to encrypt its tunnels. To demonstrate how to make DMVPN secure, we will now use IPSec and add it to the [DMVPN Phase 3](#) configuration.

### Network Architecture:

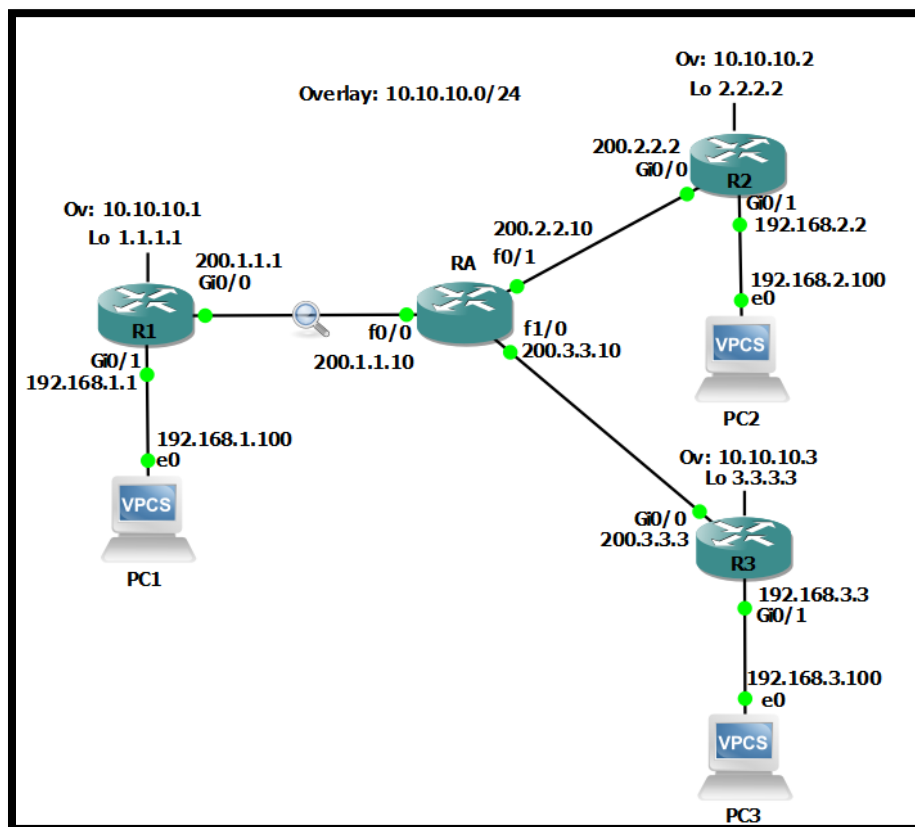


Figure 52 - DMVPN over IPSec Network Architecture

In this exercise, we maintain the same architecture as the previous laboratory exercise. Just like before, R1 is the Hub, and R2 and R3 both are Spoke routers; all these three routers are Cisco IOSv routers. The complete configuration of this lab can be found in [Annex A](#).

### Proof-of-Concept:

Using the previously configured network, our task in this exercise was a simple implementation of an IPSec policy profile to the configuration of the tunnel interface of all routers.

1. After implementing the IPSec tunnel, we analysed the routing tables and NHRP caches. Just like shown in Figure 53 and Figure 54, there are no differences compared with the application of DMVPN without IPSec.

```

Gateway of last resort is not set

    1.0.0.0/32 is subnetted, 1 subnets
C       1.1.1.1 is directly connected, Loopback0
    2.0.0.0/32 is subnetted, 1 subnets
O       2.2.2.2 [110/12] via 200.1.1.10, 00:09:53, GigabitEthernet0/0
    3.0.0.0/32 is subnetted, 1 subnets
O       3.3.3.3 [110/3] via 200.1.1.10, 00:09:53, GigabitEthernet0/0
    10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.10.10.0/24 is directly connected, Tunnel0
L       10.10.10.1/32 is directly connected, Tunnel0
    192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       192.168.1.0/24 is directly connected, GigabitEthernet0/1
L       192.168.1.1/32 is directly connected, GigabitEthernet0/1
R       192.168.2.0/24 [120/1] via 10.10.10.2, 00:00:00, Tunnel0
R       192.168.3.0/24 [120/1] via 10.10.10.3, 00:00:11, Tunnel0
    200.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       200.1.1.0/24 is directly connected, GigabitEthernet0/0
L       200.1.1.1/32 is directly connected, GigabitEthernet0/0
O       200.2.2.0/24 [110/11] via 200.1.1.10, 00:09:53, GigabitEthernet0/0
O       200.3.3.0/24 [110/2] via 200.1.1.10, 00:09:53, GigabitEthernet0/0
Router#

```

Figure 53 - R1's routing table

```

Interface: Tunnel0, IPv4 NHRP Details
Type:Hub, NHRP Peers:2,

# Ent Peer NBMA Addr Peer Tunnel Add State UpDn Tm Attrb
-----
  1 2.2.2.2          10.10.10.2    UP 00:07:42    D
  1 3.3.3.3          10.10.10.3    UP 00:07:09    D
Router#

```

Figure 54 - R1's NHRP cache

- Then, we analysed the packets going through the public network and tried to identify what packets are protected by IPSec; first, we explored an AH configuration. In this analysis, we concluded that IPSec protects all packets between the hub and spokes (overlay) but not the packets in the public network (underlay). In Figure 55 and Figure 56, we can see an NHRP Registration reply and a RIPv2 response packet, both protected by the AH header. Although in Figure 57, we can see an OSPF Hello packet unprotected by IPSec.

```

598 954.615921 1.1.1.1 2.2.2.2 NHRP 174 NHRP Registration Reply, ID=6, Code=Success
> Frame 598: 174 bytes on wire (1392 bits), 174 bytes captured (1392 bits) on interface -, id 0
> Ethernet II, Src: 0c:be:9e:79:00:00 (0c:be:9e:79:00:00), Dst: c2:01:5d:29:00:00 (c2:01:5d:29:00:00)
> Internet Protocol Version 4, Src: 1.1.1.1, Dst: 2.2.2.2
> Authentication Header
> Generic Routing Encapsulation (NHRP)
> Next Hop Resolution Protocol (NHRP Registration Reply)

```

Figure 55 - NHRP Registration Reply with IPSec AH transport mode

582	935.781793	10.10.10.3	224.0.0.9	RIPv2	114 Response
> Frame 582: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface -, id 0 > Ethernet II, Src: c2:01:5d:29:00:00 (c2:01:5d:29:00:00), Dst: 0c:be:9e:79:00:00 (0c:be:9e:79:00:00) > Internet Protocol Version 4, Src: 3.3.3.3, Dst: 1.1.1.1 > Authentication Header > Generic Routing Encapsulation (IP) > Internet Protocol Version 4, Src: 10.10.10.3, Dst: 224.0.0.9 > User Datagram Protocol, Src Port: 520, Dst Port: 520 > Routing Information Protocol					

Figure 56 - RIPv2 Response packet with IPsec AH transport mode

645	1029.996977	200.1.1.10	224.0.0.5	OSPF	94 Hello Packet
> Frame 645: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface -, id 0 > Ethernet II, Src: c2:01:5d:29:00:00 (c2:01:5d:29:00:00), Dst: IPv4mcast_05 (01:00:5e:00:00:05) > Internet Protocol Version 4, Src: 200.1.1.10, Dst: 224.0.0.5 > Open Shortest Path First					

Figure 57 - OSPF Hello packet with no IPsec protection

3. Then, we did the same for an ESP configuration. And we concluded that the same happened. As shown in Figure 58, the only packet we can identify is the OSPF Hello packets transmitted between the underlay network. Both RIPv2, NHRP and ICMP packets from the overlay network are encrypted and protected by IPsec.

No.	Time	Source	Destination	Protocol	Length	Info
26	43.423817	1.1.1.1	3.3.3.3	ESP	134	ESP (SPI=0x20473abf)
27	43.797911	200.1.1.10	224.0.0.5	OSPF	94	Hello Packet
28	44.500561	3.3.3.3	1.1.1.1	ESP	134	ESP (SPI=0x3ec12afd)
29	44.554755	0c:be:9e:79:00:00	0c:be:9e:79:00:00	LOOP	60	Reply
30	46.368208	200.1.1.1	224.0.0.5	OSPF	94	Hello Packet
31	50.000538	c2:01:5d:29:00:00	c2:01:5d:29:00:00	LOOP	60	Reply
32	53.798352	200.1.1.10	224.0.0.5	OSPF	94	Hello Packet
33	54.567236	0c:be:9e:79:00:00	0c:be:9e:79:00:00	LOOP	60	Reply
34	55.223471	1.1.1.1	2.2.2.2	ESP	166	ESP (SPI=0x19b6d5b3)
35	55.262212	2.2.2.2	1.1.1.1	ESP	166	ESP (SPI=0xd14d1b4d)
36	55.970797	200.1.1.1	224.0.0.5	OSPF	94	Hello Packet
37	56.267769	1.1.1.1	2.2.2.2	ESP	166	ESP (SPI=0x19b6d5b3)
38	56.282469	2.2.2.2	1.1.1.1	ESP	166	ESP (SPI=0xd14d1b4d)
39	57.288130	1.1.1.1	2.2.2.2	ESP	166	ESP (SPI=0x19b6d5b3)
> Frame 36: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface -, id 0 > Ethernet II, Src: 0c:be:9e:79:00:00 (0c:be:9e:79:00:00), Dst: IPv4mcast_05 (01:00:5e:00:00:05) > Internet Protocol Version 4, Src: 200.1.1.1, Dst: 224.0.0.5 > Open Shortest Path First						

Figure 58 - Capture of packets transmitted with IPsec ESP transport mode



4. We applied the transport mode in these configurations because, as previously explained in the exercise, [GRE over IPSec](#) tunnel mode uses three IP headers. In contrast, the transport mode only uses two, reducing the transmission overhead. We also analysed the SAs created between each pair of routers and concluded that there would be a new SA for each tunnel created, as shown in Figure 59.

```
Router#show crypto isakmp sa
IPv4 Crypto ISAKMP SA
dst          src          state      conn-id status
1.1.1.1      2.2.2.2      QM_IDLE    1001 ACTIVE
1.1.1.1      3.3.3.3      QM_IDLE    1002 ACTIVE
IPv6 Crypto ISAKMP SA
```

Figure 59 – R1's SAs in DMVPN over IPSec

## 2. Lab No. 2.2 – Networking of virtual containers

This lab work aims to study the networking of virtual containers, covering Linux network namespaces, Docker, and Docker Swarm. The laboratories analysed are:

- I. Connecting network namespaces to external networks
- II. Macvlan networks with VLANs
- III. Linux VxLAN with multicast routing
- IV. Docker Swarm without data encryption
- V. Docker Swarm with data encryption

### 2.1. Connecting network namespaces to external networks

Linux network namespaces are a Linux kernel feature that allows the isolation of network environments through virtualization. Using this feature allows to create separate network interfaces and routing tables that are isolated from the rest of the system. This feature is the basis of container technologies like Docker or Kubernetes. (Adams, 2021)

In this exercise, we will connect two Linux network namespaces, using a bridge interface, to external networks.

#### Network Architecture:

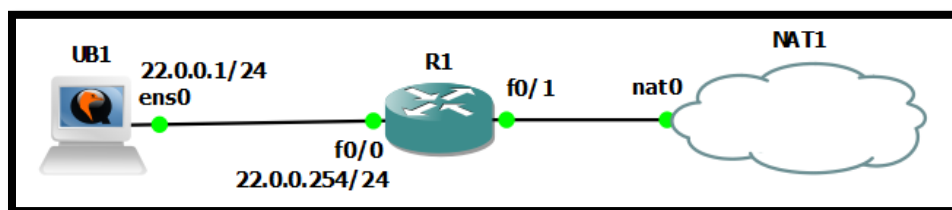


Figure 60 - Network namespaces connecting to external networks network architecture

In this network topology, we use an Ubuntu Server with a cisco router R1 as its default gateway.

#### Proof-of-Concept:

1. We started by the creation of two network namespaces red and blue identified in Figure 61

```
root@gns3:/home/gns3# ip netns
blue (id: 1)
red (id: 0)
```

Figure 61 - Active NNSes

2. Then we create a bridge interface and virtual cables and connect the bridge interface to the NNSes. These interfaces are presented in Figure 62.

```

root@gns3:/home/gns3# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
    DEFAULT group default qlen 1000
    link/ether 0c:ba:28:67:00:00 brd ff:ff:ff:ff:ff:ff
3: v-net-0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mo
    de DEFAULT group default qlen 1000
    link/ether e6:4a:ff:24:fe:61 brd ff:ff:ff:ff:ff:ff
4: veth-red-br@if5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue mas
    ter v-net-0 state UP mode DEFAULT group default qlen 1000
    link/ether ea:f9:49:89:2f:39 brd ff:ff:ff:ff:ff:ff link-netnsid 0
6: veth-blue-br@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue ma
    ster v-net-0 state UP mode DEFAULT group default qlen 1000
    link/ether e6:4a:ff:24:fe:61 brd ff:ff:ff:ff:ff:ff link-netnsid 1
root@gns3:/home/gns3#

```

Figure 62 - Bridge, red and blue interfaces

3. To connect to the external network, we configured a default gateway at the NNSes and NAT at the host so that the 11.0.0.0/24 addresses get translated into the public 22.0.0.0/24 addresses. The default gateway is the bridge's IP address, and NAT is implemented through iptables.
4. After all the configurations presented in [Annex B](#), we were able to reach the outside networks as we can see in Figure 63.

```

root@gns3:/home/gns3# ip netns exec blue ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:
64 bytes from 8.8.8.8: icmp_seq=1 ttl=125 time=27.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=125 time=27.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=125 time=22.4 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=125 time=23.4 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=125 time=28.3 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 22.457/25.861/28.381/2.409 ms

```

Figure 63 - Successful communication with external networks

## 2.2. Macvlan networks with VLANs

Macvlan is a technology that configures multiple MAC addresses on a single physical interface. Having a parent physical ethernet interface Macvlan allows the configuration of sub-interfaces of that same parent interface, attributing unique MAC and IP addresses to each sub-interface. This technology helps bind VMs or containers to a specific sub-interface or implement communication between containers located in different nodes. (Hi Cube, 2022)

Using these networks with IEEE 802.1q trunking allows mapping each Docker host interface to a macvlan network, extending the Layer 2 domain from the VLAN into the macvlan network.

### Network Architecture:

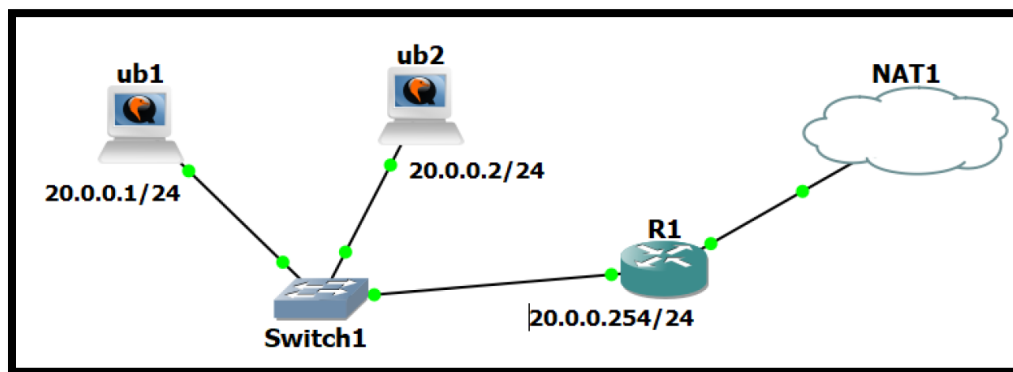


Figure 64 - Macvlan networks with VLANs Network Architecture

In this network topology, we used a 3725 router connected to NAT and a Switch that connects two Ubuntu hosts, ub1 and ub2, each with two alpine containers. The complete configuration can be found in [Annex B](#).

### Proof-of-Concept:

To complete this laboratory exercise, we created two Macvlan networks: 11.0.0.0/24 and 12.0.0.0/24. Both were configured in ub1 and ub2. For that, we used the commands:

```

1. docker network create -d macvlan --subnet=11.0.0.0/24 -o parent=ens3.11 my_8021q_net_vlan1
2. docker network create -d macvlan --subnet=12.0.0.0/24 -o parent=ens3.12 my_8021q_net_vlan2
  
```

1. The `-o` flag is used to specify the sub-interface used by each VLAN. The result of the creation of the Macvlan networks can be found in Figure 65 (for ub1, the results are similar for ub2). In Figure 66, we can observe the sub-interfaces created in ub1.

```

root@gns3:/home/gns3# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
926efd14c2f3        bridge              bridge              local
23e3b578cb62        host                host                local
4fd78a61bfae        my_8021q_net_vlan1 macvlan             local
cb9e455f7d59        my_8021q_net_vlan2 macvlan             local
baafe6ec79a6        none                null                local
root@gns3:/home/gns3#

```

Figure 65 - ub1 docker networks

```

3: I docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN g
roup default
    link/ether 02:42:42:e9:73:e2 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
4: ens3.11@ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default
    link/ether 0c:ee:a5:40:00:00 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::eee:a5ff:fe40:0/64 scope link
        valid_lft forever preferred_lft forever
5: ens3.12@ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default
    link/ether 0c:ee:a5:40:00:00 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::eee:a5ff:fe40:0/64 scope link
        valid_lft forever preferred_lft forever

```

Figure 66 - Sub-interfaces in ub1

2. We then proceeded to create two containers in both ub1 and ub2. For each host, we have a container attached to VLAN1 and a container attached to VLAN2. These containers can be found in Figure 67 and Figure 68.

```

root@gns3:/home/gns3# docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
9c0f78f78330   alpine    "/bin/sh" About a minute ago Up About a minute   cont1
5eb2f9d9f02d   alpine    "/bin/sh" About a minute ago Up About a minute   cont0
root@gns3:/home/gns3#

```

Figure 67 - Containers in ub1

```

root@gns3:/home/gns3# docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
833696937de1   alpine    "/bin/sh" 21 seconds ago Up 20 seconds   cont3
4a67f4ebc571   alpine    "/bin/sh" 40 seconds ago Up 38 seconds   cont2
root@gns3:/home/gns3#

```

Figure 68 - Containers in ub2

3. We then tested the connection with a ping from cont0 to cont2 (both attached to vlan1, with cont0 in ub1 and cont2 in ub2) and from cont1 to cont3 (both linked to vlan2, with cont1 in ub1 and cont3 in ub2). We can see in Figure 69 and in Figure 70 that both pings were successful.

```

root@gns3:/home/gns3# docker exec cont0 ping 11.0.0.12
PING 11.0.0.12 (11.0.0.12): 56 data bytes
64 bytes from 11.0.0.12: seq=0 ttl=64 time=1.775 ms
64 bytes from 11.0.0.12: seq=1 ttl=64 time=0.600 ms
64 bytes from 11.0.0.12: seq=2 ttl=64 time=0.672 ms
^C
root@gns3:/home/gns3#

```

Figure 69 - Ping from cont0 to cont2 (VLAN1)

```

root@gns3:/home/gns3# docker exec cont1 ping 12.0.0.13
PING 12.0.0.13 (12.0.0.13): 56 data bytes
64 bytes from 12.0.0.13: seq=0 ttl=64 time=1.771 ms
64 bytes from 12.0.0.13: seq=1 ttl=64 time=0.494 ms
64 bytes from 12.0.0.13: seq=2 ttl=64 time=0.485 ms
64 bytes from 12.0.0.13: seq=3 ttl=64 time=0.469 ms
^C
root@gns3:/home/gns3#

```

Figure 70 - Ping from cont1 to cont3 (VLAN2)

4. To confirm isolation, we tried to ping cont3 from cont0. The ping should not be successful since the containers are attached to different VLANs. We can ensure that cont0 could not reach cont3 by looking at Figure 71.

```

root@gns3:/home/gns3# docker exec cont0 ping 12.0.0.13
PING 12.0.0.13 (12.0.0.13): 56 data bytes
^C
root@gns3:/home/gns3#

```

Figure 71 - Ping from cont0 to cont3 (VLAN1 to VLAN2)

5. We have configured a Wireshark capture between ub1 and the switch to learn more about the communication between the containers. As we can see in Figure 72 and Figure 73, ICMP packets from both VLANs have an 802.1Q VLAN header containing the VLAN ID. VLAN ID allows to identify the source VLAN of the packet sent. Packets from containers in the same subnet will share the same VLAN ID.

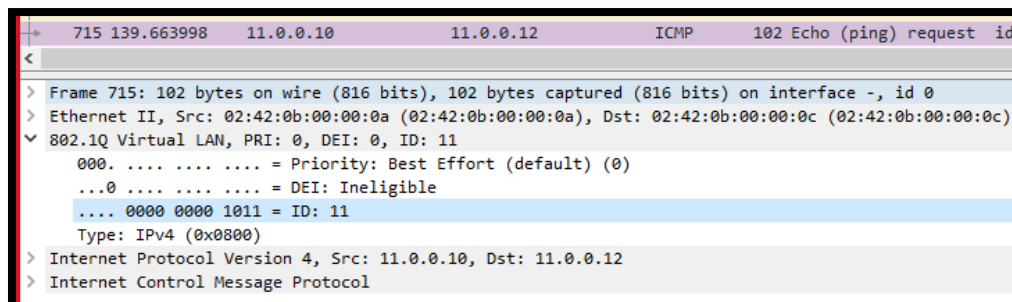


Figure 72 - ICMP packet between VLAN1 containers

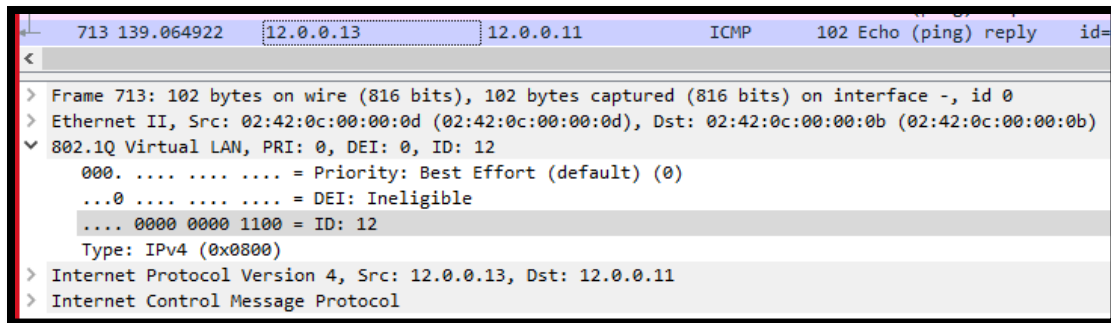


Figure 73 - ICMP packet between VLAN2 containers

### 2.3. Linux VxLAN with multicast routing

Multicast routing is used to assure connection between networks. Multicasting consists of a group of devices receiving the same messages or packets. For this process to work, multiple devices must share an IP address. Traffic sent to that IP address will reach all devices in that Multicast group. (cloudflare, 2022)

#### Network Architecture:

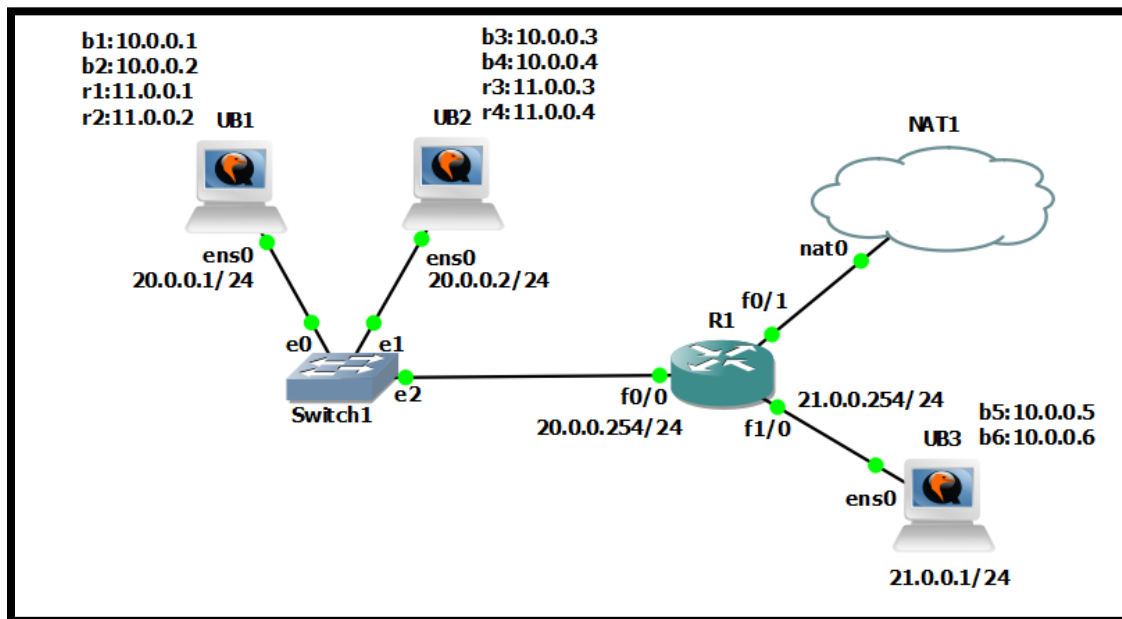


Figure 74 - Linux VxLAN with multicast routing Network Architecture

In this network topology, we use a 3725 router connected to NAT, a Ubuntu machine (UB3) and a Switch that connects two Ubuntu hosts, UB1 and UB2, each Ubuntu machine with four alpine containers except UB3 which only has two in one network (blue). The complete configuration can be found in [Annex B](#).

**Proof-of-Concept:**

1. The Internet Group Management Protocol (IGMP) will be the protocol responsible for allowing multiple devices to share the IP address. After configurations, we can see in Figure 75 the IGMP table in R1 showing the association between multicast groups and subnets.

```

R1#show ip igmp groups
IGMP Connected Group Membership
Group Address      Interface          Uptime    Expires    Last Reporter    Group Accounted
239.1.1.3          FastEthernet1/0    00:23:45  00:02:02   21.0.0.1         239.1.1.3
239.1.1.3          FastEthernet0/0    00:39:21  00:02:59   20.0.0.1         239.1.1.3
239.1.1.6          FastEthernet0/0    00:39:16  00:02:58   20.0.0.1         239.1.1.6
224.0.1.40         FastEthernet0/0    00:42:56  00:02:07   20.0.0.254       224.0.1.40

```

Figure 75 - IGMP Groups R1

2. Both UB1 and UB2 have containers attached to two different subnets: **10.0.0.0/24** and **11.0.0.0/24**. In mind, UB1 and UB2 must be part of the two multicast groups. To better explore the IGMP process, we configured a Wireshark capture between the Switch and R1 and another between R1 and UB3. The IGMP messages regarding the membership request for both UB1 and UB2 can be found in Figure 76. Here we can identify the messages with source UB1 and UB2 to join the multicast groups **230.1.1.3** attached to the VLAN blue and 239.1.1.6 connected to VLAN red.

igmp						
No.	Time	Source	Destination	Protocol	Length	Info
115	333.698818	20.0.0.1	224.0.0.251	IGMPv2	46	Membership Report group 224.0.0.251
117	334.701937	20.0.0.254	224.0.1.40	IGMPv2	60	Membership Report group 224.0.1.40
118	334.848562	20.0.0.2	239.1.1.3	IGMPv2	46	Membership Report group 239.1.1.3
128	389.697655	20.0.0.254	224.0.0.1	IGMPv2	60	Membership Query, general
131	391.243255	20.0.0.2	224.0.0.251	IGMPv2	46	Membership Report group 224.0.0.251
133	394.346443	20.0.0.1	239.1.1.3	IGMPv2	46	Membership Report group 239.1.1.3
134	395.755352	20.0.0.2	239.1.1.6	IGMPv2	46	Membership Report group 239.1.1.6
135	399.705385	20.0.0.254	224.0.1.40	IGMPv2	60	Membership Report group 224.0.1.40
> Frame 134: 46 bytes on wire (368 bits), 46 bytes captured (368 bits) on interface -, id 0 > Ethernet II, Src: 0c:ee:a5:40:00:00 (0c:ee:a5:40:00:00), Dst: IPv4mcast_01:01:06 (01:00:5e:01:01:06) > Internet Protocol Version 4, Src: 20.0.0.2, Dst: 239.1.1.6 > Internet Group Management Protocol [IGMP Version: 2] Type: Membership Report (0x16) Max Resp Time: 0.0 sec (0x00) Checksum: 0xf9f7 [correct] [Checksum Status: Good] Multicast Address: 239.1.1.6						

Figure 76 - Membership request IGMP messages of UB1 and UB2



3. Furthermore, ub3 has containers only attached to the subnet **10.0.0.0/24**. This way, UB3 only needs to be in the multicast group **239.1.1.3**. We can confirm such in Figure 77.

No.	Time	Source	Destination	Protocol	Length	Info
36	96.248462	21.0.0.1	239.1.1.3	IGMPv2	46	Membership Report group 239.1.1.3
37	97.018031	21.0.0.1	224.0.0.251	IGMPv2	46	Membership Report group 224.0.0.251
48	155.363014	21.0.0.254	224.0.0.1	IGMPv2	60	Membership Query, general
49	161.717588	21.0.0.1	239.1.1.3	IGMPv2	46	Membership Report group 239.1.1.3
51	165.046138	21.0.0.1	224.0.0.251	IGMPv2	46	Membership Report group 224.0.0.251
61	215.358282	21.0.0.254	224.0.0.1	IGMPv2	60	Membership Query, general
62	219.731579	21.0.0.1	239.1.1.3	IGMPv2	46	Membership Report group 239.1.1.3
63	224.083256	21.0.0.1	224.0.0.251	IGMPv2	46	Membership Report group 224.0.0.251
73	275.364906	21.0.0.254	224.0.0.1	IGMPv2	60	Membership Query, general
74	277.043977	21.0.0.1	224.0.0.251	IGMPv2	46	Membership Report group 224.0.0.251
76	285.194653	21.0.0.1	239.1.1.3	IGMPv2	46	Membership Report group 239.1.1.3

```

> Frame 49: 46 bytes on wire (368 bits), 46 bytes captured (368 bits) on interface -, id 0
> Ethernet II, Src: 0c:6f:43:9d:00:00 (0c:6f:43:9d:00:00), Dst: IPv4mcast_01:01:03 (01:00:5e:01:01:03)
> Internet Protocol Version 4, Src: 21.0.0.1, Dst: 239.1.1.3
  > Internet Group Management Protocol
    [IGMP Version: 2]
    Type: Membership Report (0x16)
    Max Resp Time: 0.0 sec (0x00)
    Checksum: 0xf9fa [correct]
    [Checksum Status: Good]
    Multicast Address: 239.1.1.3
  
```

Figure 77 - Membership request IGMP messages of UB3

4. In this exercise, we configured multicast routing in dense-mode. Dense mode multicast routing protocols are used for networks where most subnets in the network should receive the multicast traffic. When a router receives the multicast traffic, it will flood it on all of its interfaces except the interface where it received the multicast traffic. We can see the multicast routing table of R1 in Figure 78.

```

(*, 239.1.1.3), 01:08:24/stopped, RP 0.0.0.0, flags: DC
Incoming interface: Null, RPF nbr 0.0.0.0
Outgoing interface list:
FastEthernet1/0, Forward/Dense, 00:52:48/00:00:00
FastEthernet0/0, Forward/Dense, 01:08:24/00:00:00

(20.0.0.1, 239.1.1.3), 00:01:21/00:02:52, flags: T
Incoming interface: FastEthernet0/0, RPF nbr 0.0.0.0
Outgoing interface list:
FastEthernet1/0, Forward/Dense, 00:01:23/00:00:00

(20.0.0.2, 239.1.1.3), 00:00:13/00:02:51, flags: T
Incoming interface: FastEthernet0/0, RPF nbr 0.0.0.0
Outgoing interface list:
FastEthernet1/0, Forward/Dense, 00:00:13/00:00:00

(*, 239.1.1.6), 01:08:20/stopped, RP 0.0.0.0, flags: DC
Incoming interface: Null, RPF nbr 0.0.0.0
Outgoing interface list:
FastEthernet0/0, Forward/Dense, 01:08:20/00:00:00

(20.0.0.1, 239.1.1.6), 00:00:03/00:02:55, flags: PT
Incoming interface: FastEthernet0/0, RPF nbr 0.0.0.0
Outgoing interface list: Null

(*, 224.0.1.40), 01:11:08/00:01:54, RP 0.0.0.0, flags: DCL
Incoming interface: Null, RPF nbr 0.0.0.0
Outgoing interface list:
FastEthernet0/0, Forward/Dense, 01:11:08/00:00:00

R1#

```

Figure 78 - R1's Multicast routing table

5. We then proceed to test the connection between containers. As expected, we were able to ping between containers of the same network even if located in different hosts. Pings to containers attached to different subnets were not successful. We can see evidence of both cases in Figure 79 and Figure 80.

```

root@gns3:/home/gns3# docker exec b1 ping -c 4 10.0.0.2
PING 10.0.0.2 (10.0.0.2): 56 data bytes
64 bytes from 10.0.0.2: seq=0 ttl=64 time=5.425 ms
64 bytes from 10.0.0.2: seq=1 ttl=64 time=0.055 ms
64 bytes from 10.0.0.2: seq=2 ttl=64 time=0.057 ms
64 bytes from 10.0.0.2: seq=3 ttl=64 time=0.057 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.055/1.398/5.425 ms

```

Figure 79 - ICMP ping from b1 to b2

```

root@gns3:/home/gns3# docker exec r1 ping -c 4 10.0.0.3
PING 10.0.0.3 (10.0.0.3): 56 data bytes

--- 10.0.0.3 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss

```

Figure 80 - ICMP ping from r1 to b3

- After issuing the ping, we analysed a Wireshark capture for more information. We discovered ARP and ICMP packets generated by a ping from r1 to r4. We understood that when pinging for the first time, ARP messages are issued to map the IP associated with the destination host MAC address or, if the destination host is in a different subnet, with the default gateway. ARP and ICMP packets can be seen in Figure 81 and Figure 82 respectively.

19	55.681414	02:42:0b:00:00:01	Broadcast	ARP	92 Who has 11.0.0.4? Tell 11.0.0.1
20	55.683525	02:42:0b:00:00:04	02:42:0b:00:00:01	ARP	92 11.0.0.4 is at 02:42:0b:00:00:04

```

> Frame 20: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface -, id 0
> Ethernet II, Src: 0c:ee:a5:40:00:00 (0c:ee:a5:40:00:00), Dst: 0c:9e:8f:64:00:00 (0c:9e:8f:64:00:00)
> Internet Protocol Version 4, Src: 20.0.0.2, Dst: 20.0.0.1
> User Datagram Protocol, Src Port: 33205, Dst Port: 4789
▼ Virtual eXtensible Local Area Network
  > Flags: 0x0800, VXLAN Network ID (VNI)
    Group Policy ID: 0
    VXLAN Network Identifier (VNI): 66
    Reserved: 0
  > Ethernet II, Src: 02:42:0b:00:00:04 (02:42:0b:00:00:04), Dst: 02:42:0b:00:00:01 (02:42:0b:00:00:01)
  > Address Resolution Protocol (reply)

```

Figure 81 – ARP reply message from UB2 to UB1

No.	Time	Source	Destination	Protocol	Length	Info
21	55.684268	11.0.0.1	11.0.0.4	ICMP	148	Echo (ping) request id=0x0029, seq=0/0, ttl=64 (reply in 22)

```

> Frame 21: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface -, id 0
> Ethernet II, Src: 0c:9e:8f:64:00:00 (0c:9e:8f:64:00:00), Dst: 0c:ee:a5:40:00:00 (0c:ee:a5:40:00:00)
> Internet Protocol Version 4, Src: 20.0.0.1, Dst: 20.0.0.2
> User Datagram Protocol, Src Port: 58147, Dst Port: 4789
▼ Virtual eXtensible Local Area Network
  > Flags: 0x0800, VXLAN Network ID (VNI)
    Group Policy ID: 0
    VXLAN Network Identifier (VNI): 66
    Reserved: 0
  > Ethernet II, Src: 02:42:0b:00:00:01 (02:42:0b:00:00:01), Dst: 02:42:0b:00:00:04 (02:42:0b:00:00:04)
  > Internet Protocol Version 4, Src: 11.0.0.1, Dst: 11.0.0.4
  > Internet Control Message Protocol

```

Figure 82 - ICMP packet from r1 to r4

- We can see that both packets have a Virtual eXtensible Local Area Network (VxLAN) header. This field contains the VXLAN Network identifier (VNI). The VNI is used to identify the source VLAN of the packet sent. Furthermore, we can identify two IP headers in the ICMP packets. The outer header contains the IP of the source and destination hosts, while the inner header contains the IP of the source and destination containers.
- After the ping we can verify that the ARP cache at r1 has been updated with the IP-MAC address mapping of r4, in Figure 83.

```

root@gns3:/home/gns3# docker exec r1 arp
gns3.local (11.0.0.101) at <incomplete> on eth0
? (11.0.0.4) at 02:42:0b:00:00:04 [ether] on eth0

```

Figure 83 - ARP cache of r1

## 2.4. Docker Swarm without data encryption

Docker swarm is a container orchestration tool, allowing the user to manage multiple containers deployed across multiple machines, providing high availability for applications.

In a Docker swarm implementation, there are several **worker** nodes and at least one **manager** node responsible for efficiently handling the worker node's resources and ensuring that the cluster operates efficiently. (Sumo Logic, 2022)

All swarm service management traffic is encrypted by default, using the AES algorithm in GCM mode. Manager nodes in the swarm rotate the key used to encrypt secret data every 12 hours. A Docker swarm generates two different kinds of traffic:

- **Control and management plane traffic:** This includes swarm management messages, such as requests to join or leave the swarm. This traffic is always encrypted.
- **Application data plane traffic:** This includes container traffic and traffic to and from external clients. (Docker, 2022)

### Network Architecture:

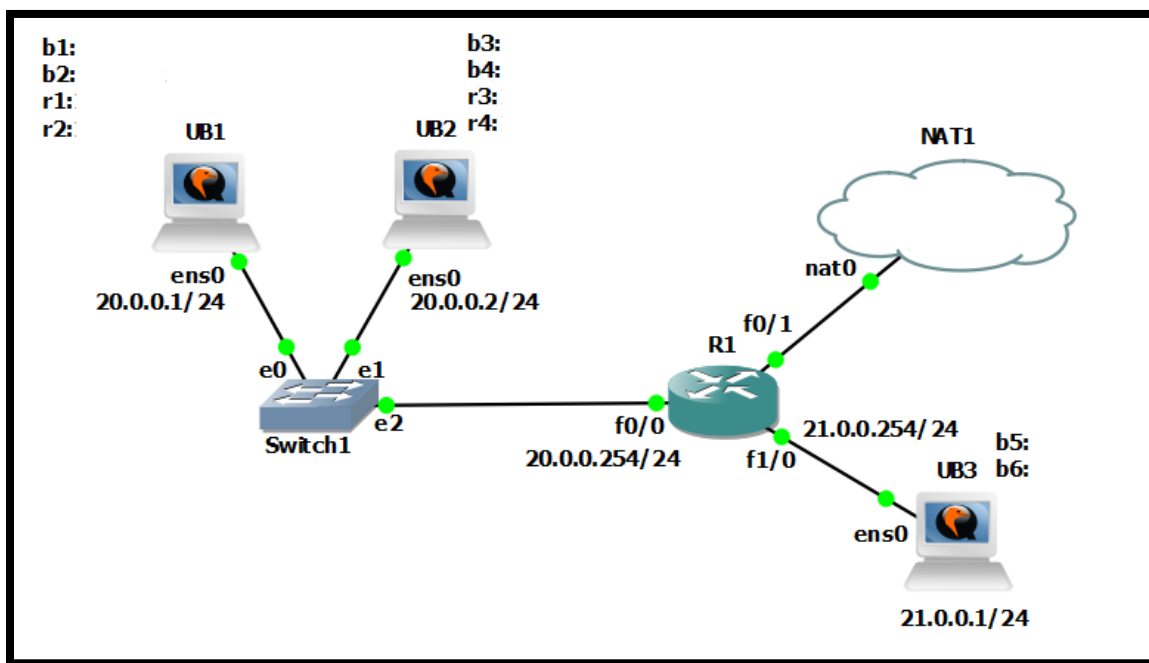


Figure 84 - Docker Swarm without data encryption network architecture

This exercise replicates the network architecture of exercise [2.3](#), but with Docker Swarm. Where UB1 is the manager and UB2 and UB3 are the workers. The complete configuration can be found in [Annex B](#).

**Proof-of-Concept:**

1. We started the configuration of this technology by assigning the roles of each node, whereas, in UB1, we generated an advertisement token and used it at the worker nodes to join them to the advertised manager. We can see all the hosts in Figure 85.

```
root@gns3:/home/gns3# docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
vtmm5812pe43jcwpxvn87izx1 *	UB1	Ready	Active	Leader	20.10.7
uj4kgknxfqxydjg73d3meal	UB2	Ready	Active		20.10.7
c3nph90jmbutzac2jwqsz9fm	UB3	Ready	Active		20.10.7

Figure 85 - Docker smarm node's roles

2. Then we created two overlay networks named blue and red, with all the required containers in Figure 84. We finalised the configuration of the network with a connectivity test of all the containers, and we can ensure the network's success in Figure 86.

```
root@gns3:/home/gns3# docker exec b1 ping -c 4 b3
PING b3 (10.0.1.7): 56 data bytes
64 bytes from 10.0.1.7: seq=0 ttl=64 time=1.336 ms
64 bytes from 10.0.1.7: seq=1 ttl=64 time=0.590 ms
64 bytes from 10.0.1.7: seq=2 ttl=64 time=0.555 ms
64 bytes from 10.0.1.7: seq=3 ttl=64 time=0.557 ms

--- b3 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.555/0.759/1.336 ms
root@gns3:/home/gns3# docker exec b1 ping -c 4 b4
PING b4 (10.0.1.9): 56 data bytes
64 bytes from 10.0.1.9: seq=0 ttl=64 time=9.234 ms
64 bytes from 10.0.1.9: seq=1 ttl=64 time=0.862 ms
64 bytes from 10.0.1.9: seq=2 ttl=64 time=0.906 ms
64 bytes from 10.0.1.9: seq=3 ttl=64 time=0.702 ms

--- b4 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.702/2.926/9.234 ms
root@gns3:/home/gns3# docker exec b2 ping -c 4 b5
PING b5 (10.0.1.10): 56 data bytes
64 bytes from 10.0.1.10: seq=0 ttl=64 time=16.287 ms
64 bytes from 10.0.1.10: seq=1 ttl=64 time=13.777 ms
64 bytes from 10.0.1.10: seq=2 ttl=64 time=16.438 ms
64 bytes from 10.0.1.10: seq=3 ttl=64 time=19.707 ms

--- b5 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 13.777/16.552/19.707 ms
root@gns3:/home/gns3# docker exec r1 ping -c 4 r4
PING r4 (10.0.2.7): 56 data bytes
64 bytes from 10.0.2.7: seq=0 ttl=64 time=14.764 ms
64 bytes from 10.0.2.7: seq=1 ttl=64 time=0.576 ms
64 bytes from 10.0.2.7: seq=2 ttl=64 time=1.237 ms
64 bytes from 10.0.2.7: seq=3 ttl=64 time=1.122 ms

--- r4 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.576/4.424/14.764 ms
root@gns3:/home/gns3#
```

Figure 86 - Connectivity test between all the containers

- Comparing the configuration of Docker Swarm with the previous lab exercise, we can conclude that Docker Swarm has a higher level of automation and simplicity. For instance, we did not have to assign IP addresses to each container or set up the VxLAN interfaces. All these configurations are done automatically. In Figure 87, we can see the details of the containers b1 and b2 at UB1, including their automatically assigned IP and MAC addresses and in Figure 88, the respective VxLAN ID of the blue network.

```
"Containers": {
  "045be051a6b30c6df3737150d19ca4eb20379e009887c6a0e01ba47f2d68edee": {
    "Name": "b1",
    "EndpointID": "ed531cfefe930041a56b74a6b1867e9af64b6a13d41a39e5515dfd1d5962a1c32",
    "MacAddress": "02:42:0a:00:01:02",
    "IPv4Address": "10.0.1.2/24",
    "IPv6Address": ""
  },
  "7679a6ada50d56f730b1eecba70df1e52ca8875654cd354731dd8b0aeec71700": {
    "Name": "b2",
    "EndpointID": "8da23f40bc740f91806b40d62dd34c536b720d4442c5cbeeac6e1184cdb0b780",
    "MacAddress": "02:42:0a:00:01:04",
    "IPv4Address": "10.0.1.4/24",
    "IPv6Address": ""
  }
},
```

Figure 87 - Blue network containers of UB1

```
"Options": {
  "com.docker.network.driver.overlay.vxlanid_list": "4097"
```

Figure 88 - Blue network VxLAN ID

Using Wireshark, we analysed the encapsulation of ICMP packets when ping between containers located in different nodes. In Figure 89 we can see an ICMP packet from b1 to b5 where we can verify that the communication uses VxLAN tunnelling and identify its VxLAN VNI as 4097. In this packet we can also confirm that the outer header contains the IP and MAC of the source and destination hosts, while the inner header contains the IP and MAC of the source and destination containers.

```

141 18.231640 10.0.1.2 10.0.1.12 ICMP 148 Echo (ping) request id=0x0012, seq=3/768, ttl=64 (reply in 142)
  > Frame 141: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface -, id 0
  > Ethernet II, Src: 0c:9e:8f:64:00:00 (0c:9e:8f:64:00:00), Dst: c2:01:06:3f:00:00 (c2:01:06:3f:00:00)
  > Internet Protocol Version 4, Src: 20.0.0.1, Dst: 21.0.0.1
  > User Datagram Protocol, Src Port: 55355, Dst Port: 4789
  > Virtual eXtensible Local Area Network
    > Flags: 0x0000, VXLAN Network ID (VNI)
      Group Policy ID: 0
      VXLAN Network Identifier (VNI): 4097
      Reserved: 0
    > Ethernet II, Src: 02:42:0a:00:01:02 (02:42:0a:00:01:02), Dst: 02:42:0a:00:01:0c (02:42:0a:00:01:0c)
    > Internet Protocol Version 4, Src: 10.0.1.2, Dst: 10.0.1.12
    > Internet Control Message Protocol

```

Figure 89 - ICMP packet from b1 to b5

- Also, while analysing the communication in the network via Wireshark we noticed that although we were pinged this container for the first time and using the container's domain names to ping, no ARP or DNS messages were being exchanged. This happens

because of the Docker Swarm control and management plane that exchanges messages, via TLS 1.2, between the containers containing the required information for communication between the different hosts to be possible.

5. We also noticed that it is impossible to see ICMP packets sent to a container of the same node because the ICMP packet reaches the destination container without exiting the node.

## 2.5. Docker Swarm with data encryption

In this lab exercise, we repeat the last configuration of exercise [2.4](#) but now with configured encryption for application data.

We add **--opt encrypted** when creating the overlay network to encrypt application data. This extra flag enables IPsec encryption at the level of the VxLAN. This encryption imposes a non-negligible performance penalty.

When overlay encryption is enabled, Docker creates IPSEC tunnels between all the nodes where tasks are scheduled for services attached to the overlay network. These tunnels also use the AES algorithm in GCM mode, and manager nodes automatically rotate the keys every 12 hours. (Docker, 2022)

### Proof-of-Concept:

We used the same topology and devices in this network as exercise [2.4](#). The devices configuration and container creation were also the same, except for the network creation commands at the manager node:

1. **#Create two overlay networks named blue and red**
2. `docker network create --opt encrypted --driver=overlay --attachable blue`
3. `docker network create --opt encrypted --driver=overlay --attachable red`

1. In the above commands, we create the same overlay networks but now with the specified **--opt encrypted** flag. We can confirm that the encrypted tunnel was built by pinging b5 from b1 (UB1 to UB3) and analysing, in Wireshark, that the ICMP packets are indeed cyphered. In Figure 90, we exhibit one of those packets where we can see the ESP protocol being applied.

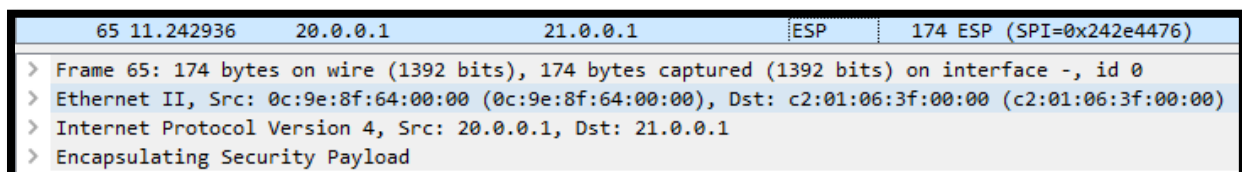


Figure 90 - Cyphered ICMP packet from b1 to b5

2. In the network configuration files we were also able to identify a new entry called encrypted below the VxLAN ID identifier, presented in Figure 91.

```
},  
"Options": {  
  "com.docker.network.driver.overlay.vxlanid_list": "4097",  
  "encrypted": ""  
},
```

Figure 91 - Encrypted entry at blue network UB1



## References:

- Adams, D. (2021). *How to Use Linux Network Namespace*. Retrieved from Linux Hint: <https://linuxhint.com/use-linux-network-namespace/#:~:text=Linux%20network%20namespaces%20are%20a,the%20system%20and%20operate%20independently.>
- Clercq, J. D. (2012, May 25). *Q: What firewall ports should we open to make IPsec work through our firewalls?* Retrieved from IT Pro Today: <https://www.itprotoday.com/strategy/q-what-firewall-ports-should-we-open-make-ipsec-work-through-our-firewalls>
- cloudflare. (2022). *What is IGMP? | Internet Group Management Protocol*. Retrieved from cloudflare: <https://www.cloudflare.com/learning/network-layer/what-is-igmp/>
- Docker. (2022). *Manage swarm service networks*. Retrieved from Docker Docs: <https://docs.docker.com/engine/swarm/networking/>
- fryadmin. (2011, August 05). *DMVPN and Routing Protocols – EIGRP*. Retrieved from Fryguy.net: <https://fryguy.net/2011/08/05/dmvpn-and-routing-protocols-eigrp/#:~:text=One%20of%20the%20joys%20of,uses%20to%20reach%20the%20destination.>
- Hi Cube. (2022). *Bridge vs Macvlan*. Retrieved from Hi Cube: <https://hicu.be/bridge-vs-macvlan>
- Network Lessons. (2022). *Introduction to DMVPN*. Retrieved from Network Lessons: <https://networklessons.com/cisco/ccie-routing-switching/introduction-to-dmvpn>
- Sumo Logic. (2022). *DevOps and Security Glossary Terms: Docker Swarm*. Retrieved from Sumo Logic: <https://www.sumologic.com/glossary/docker-swarm/#:~:text=Docker%20swarm%20is%20a%20container,of%20availability%20offered%20for%20applications.>
- Trenner, B. (2021, October 29). *Understanding Next Hop Resolution Protocol Commands*. Retrieved from Global Knowledge: [https://www.globalknowledge.com/us-en/resources/resource-library/articles/understanding-next-hop-resolution-protocol-commands/#:~:text=Next%20Hop%20Resolution%20Protocol%20\(NHRP,NHS%20is%20the%20hub%20router.](https://www.globalknowledge.com/us-en/resources/resource-library/articles/understanding-next-hop-resolution-protocol-commands/#:~:text=Next%20Hop%20Resolution%20Protocol%20(NHRP,NHS%20is%20the%20hub%20router.)

## Annex A: Configurations

### IPSec using ESP in tunnel mode

```
1. R1:
2. #Configuration
3. enable
4. conf t
5. int f0/1
6. ip address 192.168.1.1 255.255.255.0
7. no shut
8.
9. int f0/0
10. ip address 200.1.1.1 255.255.255.0
11. no shut
12. end
13.
14. #OSPF Configuration
15. conf t
16. router ospf 1
17. network 200.1.1.0 0.0.0.255 area 0
18. end
19.
20. conf t
21. ip route 192.168.2.0 255.255.255.0 200.2.2.2
22. end
23.
24. #Define interesting traffic
25. access-list 110 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
26.
27. #IKE Phase 1 - Configure ISAKMP policy
28. crypto isakmp policy 10
29. hash sha
30. authentication pre-share
31. group 5
32. lifetime 86400
33. encryption aes 256
34.
35. #IKE Phase 1 - Configure pre-shared key
36. crypto isakmp key saar address 200.2.2.2
37.
38. #IKE Phase 2 - Configure IPsec transform set
39. crypto ipsec transform-set myTSet esp-aes esp-sha-hmac
40.
41. #Configure crypto map
42. crypto map myCMap 10 ipsec-isakmp
43. set peer 200.2.2.2
44. set transform-set myTSet
45. match address 110
46.
47. #Apply crypto map to interface
48. interface f0/0
49. crypto map myCMap
50.
51. R2:
52. #Configuration
53. enable
54. conf t
55. int f0/1
56. ip address 192.168.2.2 255.255.255.0
57. no shut
58.
```

```
59. int f0/0
60. ip address 200.2.2.2 255.255.255.0
61. no shut
62.
63.
64. #OSPF Configuration
65. conf t
66. router ospf 1
67. network 200.2.2.0 0.0.0.255 area 0
68. end
69.
70. conf t
71. ip route 192.168.1.0 255.255.255.0 200.1.1.1
72. end
73.
74. #Define interesting traffic
75. access-list 110 permit ip 192.168.2.0 0.0.0.255 192.168.1.0 0.0.0.255
76.
77. #IKE Phase 1 - Configure ISAKMP policy
78. crypto isakmp policy 10
79. hash sha
80. authentication pre-share
81. group 5
82. lifetime 86400
83. encryption aes 256
84.
85. #IKE Phase 1 - Configure pre-shared key
86. crypto isakmp key saar address 200.1.1.1
87.
88. #IKE Phase 2 - Configure IPsec transform set
89. crypto ipsec transform-set myTSet esp-aes esp-sha-hmac
90.
91. #Configure crypto map
92. crypto map myCMap 10 ipsec-isakmp
93. set peer 200.1.1.1
94. set transform-set myTSet
95. match address 110
96.
97. #Apply crypto map to interface
98. interface f0/0
99. crypto map myCMap
100.
101. #Different Policy
102. #IKE Phase 1 - Configure ISAKMP policy
103. crypto isakmp policy 10
104. hash md5
105. authentication pre-share
106. group 5
107. lifetime 86400
108. encryption des
109.
110. RA:
111. #Configuration
112. enable
113. conf t
114. int f0/0
115. ip address 200.1.1.10 255.255.255.0
116. no shut
117.
118. int f0/1
119. ip address 200.2.2.10 255.255.255.0
120. no shut
121. end
122.
123. #OSPF Configuration
```

```

124. conf t
125. router ospf 1
126. network 200.1.1.0 0.0.0.255 area 0
127. network 200.2.2.0 0.0.0.255 area 0
128. end
129.
130. PC1:
131. ip 192.168.1.100 255.255.255.0 192.168.1.1
132.
133.
134. PC2:
135. ip 192.168.2.100 255.255.255.0 192.168.2.2

```

## IPSec with digital certificates and certificate authority

```

1. R1:
2. #Configuration
3. enable
4. conf t
5. int g0/1
6. ip address 192.168.1.1 255.255.255.0
7. no shut
8.
9. int g0/0
10. ip address 200.1.1.1 255.255.255.0
11. no shut
12. end
13.
14. #OSPF Configuration
15. conf t
16. router ospf 1
17. network 200.1.1.0 0.0.0.255 area 0
18. end
19.
20. conf t
21. ip route 192.168.2.0 255.255.255.0 200.2.2.2
22. end
23.
24. #CA Client Configuration
25. crypto pki trustpoint myTrustpoint
26. enrollment url http://200.1.1.10:80
27. exit
28.
29. crypto ca authenticate myTrustpoint
30.
31. crypto ca enroll myTrustpoint
32. show crypto pki certificates
33. show crypto key mypub rsa
34.
35. #Define interesting traffic
36. access-list 110 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
37.
38. #IKE Phase 1 - Configure ISAKMP policy
39. crypto isakmp policy 10
40. hash sha
41. authentication rsa-sig
42. group 5
43. lifetime 86400
44. encryption aes 256
45.
46. #IKE Phase 2 - Configure IPsec transform set
47. crypto ipsec transform-set myTSet esp-aes esp-sha-hmac

```

```
48.
49. #Configure crypto map
50. crypto map myCMap 10 ipsec-isakmp
51. set peer 200.2.2.2
52. set transform-set myTSet
53. match address 110
54.
55. #Apply crypto map to interface
56. interface g0/0
57. crypto map myCMap
58.
59. R2:
60. #Configuration
61. enable
62. conf t
63. int g0/1
64. ip address 192.168.2.2 255.255.255.0
65. no shut
66.
67. int g0/0
68. ip address 200.2.2.2 255.255.255.0
69. no shut
70. end
71.
72. #OSPF Configuration
73. conf t
74. router ospf 1
75. network 200.2.2.0 0.0.0.255 area 0
76. end
77.
78. conf t
79. ip route 192.168.1.0 255.255.255.0 200.1.1.1
80. end
81.
82. #CA Client Configuration
83. conf t
84. crypto pki trustpoint myTrustpoint
85. enrollment url http://200.1.1.10:80
86. exit
87.
88. crypto ca authenticate myTrustpoint
89.
90. crypto ca enroll myTrustpoint
91.
92. #Define interesting traffic
93. access-list 110 permit ip 192.168.2.0 0.0.0.255 192.168.1.0 0.0.0.255
94.
95. #IKE Phase 1 - Configure ISAKMP policy
96. crypto isakmp policy 10
97. hash sha
98. authentication rsa-sig
99. group 5
100. lifetime 86400
101. encryption aes 256
102.
103. #IKE Phase 2 - Configure IPsec transform set
104. crypto ipsec transform-set myTSet esp-aes esp-sha-hmac
105.
106. #Configure crypto map
107. crypto map myCMap 10 ipsec-isakmp
108. set peer 200.1.1.1
109. set transform-set myTSet
110. match address 110
111.
112. #Apply crypto map to interface
```

```
113. interface g0/0
114. crypto map myCMap
115.
116. RA:
117. #Configuration
118. enable
119. conf t
120. int f0/0
121. ip address 200.1.1.10 255.255.255.0
122. no shut
123.
124. int f0/1
125. ip address 200.2.2.10 255.255.255.0
126. no shut
127. end
128.
129. #OSPF Configuration
130. conf t
131. router ospf 1
132. network 200.1.1.0 0.0.0.255 area 0
133. network 200.2.2.0 0.0.0.255 area 0
134. end
135.
136. #CA Configuration
137. conf t
138. ip http server
139. crypto pki server myCA
140. issuer-name cn=saarCA
141. lifetime certificate 365
142. grant auto
143. no shutdown
144. saarlab2
145.
146. show crypto pki server
147. show crypto ca certificate
148. show crypto key mypub rsa
149.
150.
151. PC1:
152. ip 192.168.1.100 255.255.255.0 192.168.1.1
153.
154.
155. PC2:
156. ip 192.168.2.100 255.255.255.0 192.168.2.2
```

## IPSec with NAT traversal

```
1. R1:
2. #Configuration
3. enable
4. conf t
5. int f0/1
6. ip address 192.168.1.1 255.255.255.0
7. no shut
8.
9. int f0/0
10. ip address 192.168.3.1 255.255.255.0
11. no shut
12. end
13.
14. #OSPF R1 CONFIGURATION
15. conf t
16. router ospf 1
17. network 192.168.1.0 0.0.0.255 area 0
18. network 192.168.3.0 0.0.0.255 area 0
19. end
20.
21. #Define interesting traffic
22. conf t
23. access-list 110 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
24.
25. #IKE Phase 1 - Configure ISAKMP policy
26. crypto isakmp policy 10
27. encryption aes 256
28. authentication pre-share
29. group 5
30. exit
31.
32. #IKE Phase 1 - Configure pre-shared key
33. crypto isakmp key saar address 200.2.2.2
34.
35. #IKE Phase 2 - Configure IPsec transform set
36. crypto ipsec transform-set myTSet esp-aes esp-sha-hmac
37.
38. #Configure crypto map
39. crypto map myCMap 10 ipsec-isakmp
40. set peer 200.2.2.2
41. set transform-set myTSet
42. match address 110
43.
44. #Apply crypto map to interface
45. interface f0/0
46. crypto map myCMap
47.
48. R2:
49. #Configuration
50. enable
51. conf t
52. int f0/1
53. ip address 192.168.2.2 255.255.255.0
54. no shut
55.
56. int f0/0
57. ip address 200.2.2.2 255.255.255.0
58. no shut
59. end
60.
61. #Default route
```

```
62. conf t
63. ip route 0.0.0.0 0.0.0.0 200.2.2.10
64. end
65.
66. #Define interesting traffic
67. conf t
68. access-list 110 permit ip 192.168.2.0 0.0.0.255 192.168.1.0 0.0.0.255
69.
70. #IKE Phase 1 - Configure ISAKMP policy
71. crypto isakmp policy 10
72. encryption aes 256
73. authentication pre-share
74. group 5
75. exit
76.
77. #IKE Phase 1 - Configure pre-shared key
78. crypto isakmp key saar address 200.2.2.10
79.
80. #IKE Phase 2 - Configure IPsec transform set
81. crypto ipsec transform-set myTSet esp-aes esp-sha-hmac
82.
83. #Configure crypto map
84. crypto map myCMap 10 ipsec-isakmp
85. set peer 200.2.2.10
86. set transform-set myTSet
87. match address 110
88.
89. #Apply crypto map to interface
90. interface f0/0
91. crypto map myCMap
92. end
93.
94. RA:
95. #Configuration
96. enable
97. conf t
98. int f0/1
99. ip address 200.2.2.10 255.255.255.0
100. no shut
101.
102. int f0/0
103. ip address 192.168.3.10 255.255.255.0
104. no shut
105. end
106.
107. #OSPF AND DEFAULT ROUTER CONFIGURATION
108. conf t
109. router ospf 1
110. network 192.168.3.0 0.0.0.255 area 0
111. default-information originate
112. ip route 0.0.0.0 0.0.0.0 200.2.2.2
113. end
114.
115. #NAT CONFIGURATION
116. conf t
117. access-list 1 permit 192.168.1.0 0.255.255.255
118. access-list 1 permit 192.168.3.0 0.255.255.255
119. ip nat inside source list 1 interface f0/1 overload
120. int f0/0
121. ip nat inside
122. int f0/1
123. ip nat outside
124. end
125.
126.
```



```
127. PC1:
128. ip 192.168.1.100 255.255.255.0 192.168.1.1
129.
130.
131. PC2:
132. ip 192.168.2.100 255.255.255.0 192.168.2.2
```

## GRE over IPSec

```
1. R1:
2. #Configuration
3. enable
4. conf t
5. int f0/1
6. ip address 192.168.1.1 255.255.255.0
7. no shut
8.
9. int f0/0
10. ip address 200.1.1.1 255.255.255.0
11. no shut
12.
13. int loopback0
14. ip address 1.1.1.1 0.0.0.0
15. no shut
16. end
17.
18. #OSPF Configuration
19. conf t
20. router ospf 1
21. network 1.1.1.1 0.0.0.0 area 0
22. network 200.1.1.0 0.0.0.255 area 0
23. end
24.
25. router ospf 2
26. router-id 1.1.1.1
27.
28. #IKE Phase 1 - Configure ISAKMP policy
29. conf t
30. crypto isakmp policy 10
31. encryption aes 256
32. authentication pre-share
33. group 5
34.
35. #Configure pre-shared key
36. crypto isakmp key saar address 2.2.2.2
37.
38. #IKE Phase 2 - Configure IPsec transform set - AH
39. crypto ipsec transform-set myTSet ah-sha-hmac
40.
41. #IKE Phase 2 - Configure IPsec transform set - ESP
42. crypto ipsec transform-set myTSet esp-aes esp-sha-hmac
43.
44. #IKE Phase 2 - Configure IPsec transform set - AH Transport
45. crypto ipsec transform-set myTSet ah-sha-hmac
46. mode transport
47.
48. #Configure ipsec profile
49. crypto ipsec profile myIPSecProfile
50. set transform-set myTSet
51.
52. #Configure tunnel interface
53. interface tunnel 0
```

```
54. ip unnumbered f0/1
55. tunnel source Loopback0
56. tunnel destination 2.2.2.2
57. tunnel mode gre ip
58. tunnel protection ipsec profile myIPSecProfile
59. ip ospf 2 area 0
60.
61. #Assign OSPF process 2 to interface
62. int f0/1
63. ip ospf 2 area 0
64.
65.
66. R2:
67. #Configuration
68. enable
69. conf t
70. int f0/1
71. ip address 192.168.2.2 255.255.255.0
72. no shut
73.
74. int f0/0
75. ip address 200.2.2.2 255.255.255.0
76. no shut
77.
78. int loopback0
79. ip address 2.2.2.2 0.0.0.0
80. no shut
81. end
82.
83. #OSPF Configuration
84. conf t
85. router ospf 1
86. network 2.2.2.2 0.0.0.0 area 0
87. network 200.2.2.0 0.0.0.255 area 0
88. end
89.
90. router ospf 2
91. router-id 2.2.2.2
92.
93. #IKE Phase 1 - Configure ISAKMP policy
94. conf t
95. crypto isakmp policy 10
96. encryption aes 256
97. authentication pre-share
98. group 5
99.
100. #Configure pre-shared key
101. crypto isakmp key saar address 1.1.1.1
102.
103. #IKE Phase 2 - Configure IPsec transform set - AH
104. crypto ipsec transform-set myTSet ah-sha-hmac
105.
106. #IKE Phase 2 - Configure IPsec transform set - ESP
107. crypto ipsec transform-set myTSet esp-aes esp-sha-hmac
108.
109. #IKE Phase 2 - Configure IPsec transform set - AH Transport
110. crypto ipsec transform-set myTSet ah-sha-hmac
111. mode transport
112.
113. #Configure ipsec profile
114. crypto ipsec profile myIPSecProfile
115. set transform-set myTSet
116.
117. #Configure tunnel interface
118. interface tunnel 0
```

```
119. ip unnumbered f0/1
120. tunnel source Loopback0
121. tunnel destination 1.1.1.1
122. tunnel mode gre ip
123. tunnel protection ipsec profile myIPSecProfile
124. ip ospf 2 area 0
125.
126. #Assign OSPF process 2 to interface
127. int f0/1
128. ip ospf 2 area 0
129.
130. RA:
131. #Configuration
132. enable
133. conf t
134. int f0/0
135. ip address 200.1.1.10 255.255.255.0
136. no shut
137.
138. int f1/1
139. ip address 200.2.2.10 255.255.255.0
140. no shut
141. end
142.
143. #OSPF Configuration
144. conf t
145. router ospf 1
146. network 200.1.1.0 0.0.0.255 area 0
147. network 200.2.2.0 0.0.0.255 area 0
148. end
149.
150. PC1:
151. ip 192.168.1.100 255.255.255.0 192.168.1.1
152.
153.
154. PC2:
155. ip 192.168.2.100 255.255.255.0 192.168.2.2
156.
```

**DMVPN Phase 3**

```
1. R1:
2. #Configuration
3. enable
4. conf t
5. int gi0/1
6. ip address 192.168.1.1 255.255.255.0
7. no shut
8.
9. int gi0/0
10. ip address 200.1.1.1 255.255.255.0
11. no shut
12.
13. int loopback0
14. ip address 1.1.1.1 255.255.255.255
15. no shut
16. end
17.
18. #OSPF Configuration
19. conf t
20. router ospf 1
21. router-id 1.1.1.1
22. network 200.1.1.0 0.0.0.255 area 0
23. network 1.1.1.1 0.0.0.0 area 0
24. end
25.
26. #RIPv2 configuration
27. conf t
28. router rip
29. version 2
30. no auto-summary
31. network 192.168.1.0
32. network 10.10.10.0
33. end
34.
35. #Configuration of the tunnel interface
36. conf t
37. int Tunnel0
38. ip add 10.10.10.1 255.255.255.0
39. tunnel source Lo0
40. tunnel mode gre multipoint
41. ip nhrp network-id 1
42. ip nhrp map multicast dynamic
43. ip nhrp redirect
44. ip summary-address rip 0.0.0.0 0.0.0.0
45. end
46.
47. no ip summary-address rip 0.0.0.0 0.0.0.0
48. no ip split-horizon
49. show dmvpn
50.
51. R2:
52. #Configuration
53. enable
54. conf t
55. int g0/1
56. ip address 192.168.2.2 255.255.255.0
57. no shut
58.
59. int g0/0
60. ip address 200.2.2.2 255.255.255.0
61. no shut
```

```
62.
63. int loopback0
64. ip address 2.2.2.2 255.255.255.255
65. no shut
66. end
67.
68. #OSPF Configuration
69. conf t
70. router ospf 1
71. router-id 2.2.2.2
72. network 200.2.2.0 0.0.0.255 area 0
73. network 2.2.2.2 0.0.0.0 area 0
74. end
75.
76. #RIPv2 configuration
77. conf t
78. router rip
79. version 2
80. no auto-summary
81. network 192.168.2.0
82. network 10.10.10.0
83. end
84.
85. #Configuration of the tunnel interface
86. conf t
87. int Tunnel0
88. ip add 10.10.10.2 255.255.255.0
89. tunnel source Lo0
90. tunnel mode gre multipoint
91. ip nhrp map 10.10.10.1 1.1.1.1
92. ip nhrp map multicast 1.1.1.1
93. ip nhrp nhs 10.10.10.1
94. ip nhrp network-id 1
95. ip nhrp shortcut
96. exit
97.
98. R3:
99. #Configuration
100. enable
101. conf t
102. int g0/1
103. ip address 192.168.3.3 255.255.255.0
104. no shut
105.
106. int g0/0
107. ip address 200.3.3.3 255.255.255.0
108. no shut
109.
110. int loopback0
111. ip address 3.3.3.3 255.255.255.255
112. no shut
113. end
114.
115. #OSPF Configuration
116. conf t
117. router ospf 1
118. router-id 3.3.3.3
119. network 200.3.3.0 0.0.0.255 area 0
120. network 3.3.3.3 0.0.0.0 area 0
121. end
122.
123. #RIPv2 configuration
124. conf t
125. router rip
126. version 2
```

```
127. no auto-summary
128. network 192.168.3.0
129. network 10.10.10.0
130. end
131.
132. #Configuration of the tunnel interface
133. conf t
134. int Tunnel0
135. ip add 10.10.10.3 255.255.255.0
136. tunnel source Lo0
137. tunnel mode gre multipoint
138. tunnel protection ipsec profile myIPSecProfile
139. ip nhrp map 10.10.10.1 1.1.1.1
140. ip nhrp map multicast 1.1.1.1
141. ip nhrp nhs 10.10.10.1
142. ip nhrp network-id 1
143. ip nhrp shortcut
144. end
145.
146. RA:
147. #Configuration
148. enable
149. conf t
150. int f0/0
151. ip address 200.1.1.10 255.255.255.0
152. no shut
153.
154. int f0/1
155. ip address 200.2.2.10 255.255.255.0
156. no shut
157.
158. int f1/0
159. ip address 200.3.3.10 255.255.255.0
160. no shut
161. end
162.
163. #OSPF Configuration
164. conf t
165. router ospf 1
166. network 200.1.1.0 0.0.0.255 area 0
167. network 200.2.2.0 0.0.0.255 area 0
168. network 200.3.3.0 0.0.0.255 area 0
169. end
170.
171. PC1:
172. ip 192.168.1.100 255.255.255.0 192.168.1.1
173.
174.
175. PC2:
176. ip 192.168.2.100 255.255.255.0 192.168.2.2
177.
178.
179. PC3:
180. ip 192.168.3.100 255.255.255.0 192.168.3.3
181.
```

**DMVPN over IPSec**

```
1. R1:
2. #Configuration
3. enable
4. conf t
5. int gi0/1
6. ip address 192.168.1.1 255.255.255.0
7. no shut
8.
9. int gi0/0
10. ip address 200.1.1.1 255.255.255.0
11. no shut
12.
13. int loopback0
14. ip address 1.1.1.1 255.255.255.255
15. no shut
16. end
17.
18. #RIPv2 configuration
19. conf t
20. router rip
21. version 2
22. no auto-summary
23. network 192.168.1.0
24. network 10.10.10.0
25. end
26.
27. #OSPF Configuration
28. conf t
29. router ospf 1
30. router-id 1.1.1.1
31. network 200.1.1.0 0.0.0.255 area 0
32. network 1.1.1.1 0.0.0.0 area 0
33. end
34.
35. #IKE Phase 1 - Configure ISAKMP policy
36. conf t
37. crypto isakmp policy 10
38. encryption aes 256
39. authentication pre-share
40. group 5
41.
42. #Configure pre-shared key
43. crypto isakmp key saar address 0.0.0.0
44.
45. #IKE Phase 2 - Configure IPsec transform set - AH Transport
46. crypto ipsec transform-set myTSet ah-sha-hmac
47. mode transport
48.
49. #IKE Phase 2 - Configure IPsec transform set
50. crypto ipsec transform-set myTSet esp-aes esp-sha-hmac
51. mode transport
52.
53. #Configure ipsec profile
54. crypto ipsec profile myIPSecProfile
55. set transform-set myTSet
56. exit
57.
58. #Configuration of the tunnel interface
59. conf t
60. int Tunnel0
61. ip add 10.10.10.1 255.255.255.0
```

```
62. tunnel source Lo0
63. tunnel mode gre multipoint
64. tunnel protection ipsec profile myIPSecProfile
65. ip nhrp network-id 1
66. ip nhrp map multicast dynamic
67. ip nhrp redirect
68. ip summary-address rip 0.0.0.0 0.0.0.0
69. end
70.
71. R2:
72. #Configuration
73. enable
74. conf t
75. int g0/1
76. ip address 192.168.2.2 255.255.255.0
77. no shut
78.
79. int g0/0
80. ip address 200.2.2.2 255.255.255.0
81. no shut
82.
83. int loopback0
84. ip address 2.2.2.2 255.255.255.255
85. no shut
86. end
87.
88. #RIPv2 configuration
89. conf t
90. router rip
91. version 2
92. no auto-summary
93. network 192.168.2.0
94. network 10.10.10.0
95. end
96.
97. #OSPF Configuration
98. conf t
99. router ospf 1
100. router-id 2.2.2.2
101. network 200.2.2.0 0.0.0.255 area 0
102. network 2.2.2.2 0.0.0.0 area 0
103. end
104.
105. #IKE Phase 1 - Configure ISAKMP policy
106. conf t
107. crypto isakmp policy 10
108. encryption aes 256
109. authentication pre-share
110. group 5
111.
112. #Configure pre-shared key
113. crypto isakmp key saar address 0.0.0.0
114.
115. #IKE Phase 2 - Configure IPsec transform set - AH Transport
116. crypto ipsec transform-set myTSet ah-sha-hmac
117. mode transport
118.
119. #IKE Phase 2 - Configure IPsec transform set
120. crypto ipsec transform-set myTSet esp-aes esp-sha-hmac
121. mode transport
122.
123. #Configure ipsec profile
124. crypto ipsec profile myIPSecProfile
125. set transform-set myTSet
126.
```



```
127. #Configuration of the tunnel interface
128. conf t
129. int Tunnel0
130. ip add 10.10.10.2 255.255.255.0
131. tunnel source Lo0
132. tunnel mode gre multipoint
133. tunnel protection ipsec profile myIPSecProfile
134. ip nhrp map 10.10.10.1 1.1.1.1
135. ip nhrp map multicast 1.1.1.1
136. ip nhrp nhs 10.10.10.1
137. ip nhrp network-id 1
138. ip nhrp shortcut
139. exit
140.
141. R3:
142. #Configuration
143. enable
144. conf t
145. int g0/1
146. ip address 192.168.3.3 255.255.255.0
147. no shut
148.
149. int g0/0
150. ip address 200.3.3.3 255.255.255.0
151. no shut
152.
153. int loopback0
154. ip address 3.3.3.3 255.255.255.255
155. no shut
156. end
157.
158. #RIPv2 configuration
159. conf t
160. router rip
161. version 2
162. no auto-summary
163. network 192.168.3.0
164. network 10.10.10.0
165. end
166.
167. #OSPF Configuration
168. conf t
169. router ospf 1
170. router-id 3.3.3.3
171. network 200.3.3.0 0.0.0.255 area 0
172. network 3.3.3.3 0.0.0.0 area 0
173. end
174.
175. #IKE Phase 1 - Configure ISAKMP policy
176. conf t
177. crypto isakmp policy 10
178. encryption aes 256
179. authentication pre-share
180. group 5
181.
182. #Configure pre-shared key
183. crypto isakmp key saar address 0.0.0.0
184.
185. #IKE Phase 2 - Configure IPsec transform set - AH Transport
186. crypto ipsec transform-set myTSet ah-sha-hmac
187. mode transport
188.
189. #IKE Phase 2 - Configure IPsec transform set
190. crypto ipsec transform-set myTSet esp-aes esp-sha-hmac
191. mode transport
```

```
192.
193. #Configure ipsec profile
194. crypto ipsec profile myIPSecProfile
195. set transform-set myTSet
196.
197. #Configuration of the tunnel interface
198. conf t
199. int Tunnel0
200. ip add 10.10.10.3 255.255.255.0
201. tunnel source Lo0
202. tunnel mode gre multipoint
203. tunnel protection ipsec profile myIPSecProfile
204. ip nhrp map 10.10.10.1 1.1.1.1
205. ip nhrp map multicast 1.1.1.1
206. ip nhrp nhs 10.10.10.1
207. ip nhrp network-id 1
208. ip nhrp shortcut
209.
210. RA:
211. #Configuration
212. enable
213. conf t
214. int f0/0
215. ip address 200.1.1.10 255.255.255.0
216. no shut
217.
218. int f0/1
219. ip address 200.2.2.10 255.255.255.0
220. no shut
221.
222. int f1/0
223. ip address 200.3.3.10 255.255.255.0
224. no shut
225. end
226.
227. #OSPF Configuration
228. conf t
229. router ospf 1
230. network 200.1.1.0 0.0.0.255 area 0
231. network 200.2.2.0 0.0.0.255 area 0
232. network 200.3.3.0 0.0.0.255 area 0
233. end
234.
235. PC1:
236. ip 192.168.1.100 255.255.255.0 192.168.1.1
237.
238.
239. PC2:
240. ip 192.168.2.100 255.255.255.0 192.168.2.2
241.
242.
243. PC3:
244. ip 192.168.3.100 255.255.255.0 192.168.3.3
245.
```

## Annex B: Configurations

### Connecting network namespaces to external networks

```
1. R1 Configuration:
2.
3. conf t
4. int f0/0
5. ip add 22.0.0.254 255.255.255.0
6. ip nat inside
7. no shut
8.
9. int f0/1
10. ip add dhcp
11. ip nat outside
12. no shut
13. exit
14. ip nat inside source list 1 interface FastEthernet0/1 overload
15.
16. conf t
17. access-list 1 permit 22.0.0.0 0.0.0.255
18. end
19.
20. Ubuntu 1 Configuration:
21. sudo ifconfig ens3 22.0.0.1/24
22. sudo ip route add default via 22.0.0.254
23.
24. #Two Network Namespaces
25. ip netns add red
26. ip netns add blue
27. ip netns
28.
29. #Create network bridge interface
30. ip link add v-net-0 type bridge
31. ip link set dev v-net-0 up
32. ip link
33.
34. #Create cables
35. ip link add veth-red type veth peer name veth-red-br
36. ip link add veth-blue type veth peer name veth-blue-br
37.
38. #Assign the "cables" to the interfaces
39. ip link set veth-red netns red
40. ip link set veth-red-br master v-net-0
41.
42. ip link set veth-blue netns blue
43. ip link set veth-blue-br master v-net-0
44. ip link
45.
46. #Assign IP addresses to interfaces
47. ip -n red addr add 11.0.0.1/24 dev veth-red
48. ip -n blue addr add 11.0.0.2/24 dev veth-blue
49.
50. #Bring interfaces up
51. ip -n red link set veth-red up
52. ip -n blue link set veth-blue up
53. ip link set veth-red-br up
54. ip link set veth-blue-br up
55.
56. #Check connection
57. ip netns exec red ping 11.0.0.2
58. ping 11.0.0.1
```

```

59.
60. #Assign an IP address to the bridge interface
61. ip addr add 11.0.0.25/24 dev v-net-0
62. ip add
63. ping 11.0.0.1
64.
65. #Configure a default route at red and blue
66. ip netns exec red ip route add default via 11.0.0.25
67. ip netns exec blue ip route add default via 11.0.0.25
68.
69. #IPtables NAT implementation
70. iptables -t nat -A POSTROUTING -s 11.0.0.0/24 -o ens3 -j MASQUERADE
71. sudo sysctl -w net.ipv4.ip_forward=1
72.
73. iptables -L
74. iptables -L -v -n
75. iptables -t nat -L
76. iptables -t nat -v -x -n -L
77. iptables -t nat -F
78.
79. #Communicate with external networks
80. ip netns exec blue ping 8.8.8.8
81.

```

## Macvlan networks with VLANs

```

1. R1 Configuration:
2.
3. conf t
4. int f0/0
5. ip add 20.0.0.254 255.255.255.0
6. ip nat inside
7. no shut
8.
9. int f0/1
10. ip add dhcp
11. ip nat outside
12. no shut
13. exit
14. ip nat inside source list 1 interface FastEthernet0/1 overload
15. end
16.
17. conf t
18. access-list 1 permit 20.0.0.0 0.0.0.255
19. end
20.
21. Ubuntu 1 Configuration:
22. sudo ifconfig ens3 20.0.0.1/24
23. sudo ip route add default via 20.0.0.254
24.
25. #Docker Installation
26. apt-get update
27. apt install docker.io
28. reboot
29. systemctl status docker
30. ip link
31.
32. #Create 802.1q macvlan networks
33. docker network create -d macvlan --subnet=11.0.0.0/24 -o parent=ens3.11 my_8021q_net_vlan1
34. docker network create -d macvlan --subnet=12.0.0.0/24 -o parent=ens3.12 my_8021q_net_vlan2
35. docker network ls
36. ip addr show

```

```
37.
38. #Create and run two alpine containers
39. docker run --net=my_8021q_net_vlan1 --ip=11.0.0.10 --name=cont0 -itd alpine /bin/sh
40. docker run --net=my_8021q_net_vlan2 --ip=12.0.0.11 --name=cont1 -itd alpine /bin/sh
41. docker container ls
42. docker ps
43.
44. docker exec cont0 ping 20.0.0.13
45.
46. Ubuntu 2 Configuration:
47. sudo ifconfig ens3 20.0.0.2/24
48. sudo ip route add default via 20.0.0.254
49.
50. #Docker Installation
51. apt-get update
52. apt install docker.io
53. reboot
54. systemctl status docker
55. ip link
56.
57. #Create a 802.1q macvlan network
58. docker network create -d macvlan --subnet=11.0.0.0/24 -o parent=ens3.11 my_8021q_net_vlan1
59. docker network create -d macvlan --subnet=12.0.0.0/24 -o parent=ens3.12 my_8021q_net_vlan2
60. docker network ls
61. ip addr show
62.
63. #Create and run two alpine containers
64. docker run --net=my_8021q_net_vlan1 --ip=11.0.0.12 --name=cont2 -itd alpine /bin/sh
65. docker run --net=my_8021q_net_vlan2 --ip=12.0.0.13 --name=cont3 -itd alpine /bin/sh
66. docker container ls
67. docker ps
68.
69. #Remove Macvlans
70. docker network rm my_8021q_net_vlan1
71. docker network rm my_8021q_net_vlan2
72.
```

## Linux VxLAN with multicast routing

```

1. R1 Configuration:
2.
3. conf t
4. ip multicast-routing
5. int f0/0
6. ip add 20.0.0.254 255.255.255.0
7. ip nat inside
8. ip pim dense-mode
9. no shut
10.
11. conf t
12. int f1/0
13. ip add 21.0.0.254 255.255.255.0
14. ip nat inside
15. ip pim dense-mode
16. no shut
17.
18. int f0/1
19. ip add dhcp
20. ip nat outside
21. no shut
22. exit
23. ip nat inside source list 1 interface FastEthernet0/1 overload
24. end
25.
26. conf t
27. access-list 1 permit 20.0.0.0 0.0.0.255
28. access-list 1 permit 21.0.0.0 0.0.0.255
29. end
30.
31. #Test
32. show ip igmp groups
33. show ip mroute
34.
35. Ubuntu 1 Configuration:
36. sudo ifconfig ens3 20.0.0.1/24
37. sudo ip route add default via 20.0.0.254
38.
39. #Create blue and red VxLAN interfaces
40. ip link add blue type vxlan id 33 group 239.1.1.3 ttl 5 dev ens3 dstport 4789
41. ip link add red type vxlan id 66 group 239.1.1.6 ttl 5 dev ens3 dstport 4789
42.
43. #Activate the VxLAN interfaces
44. ip link set blue up
45. ip link set red up
46.
47. #Assign IP addresses to the VxLAN interfaces
48. ip addr add 10.0.0.101/24 dev blue
49. ip addr add 11.0.0.101/24 dev red
50.
51. #Create two macvlan networks called bluenet and rednet
52. docker network create -d macvlan --subnet=10.0.0.0/24 --gateway=10.0.0.101 -o parent=blue bluenet
53. docker network create -d macvlan --subnet=11.0.0.0/24 --gateway=11.0.0.101 -o parent=red rednet
54.
55. #Create two alpine containers and attach them to bluenet
56. docker run --privileged --net=bluenet --name=b1 --ip=10.0.0.1 -itd alpine /bin/sh
57. docker run --privileged --net=bluenet --name=b2 --ip=10.0.0.2 -itd alpine /bin/sh
58.
59. #Create two alpine containers and attach them to rednet

```

```

60. docker run --privileged --net=rednet --name=r1 --ip=11.0.0.1 -itd alpine /bin/sh
61. docker run --privileged --net=rednet --name=r2 --ip=11.0.0.2 -itd alpine /bin/sh
62.
63. #Test
64. docker exec b1 ping -c 4 10.0.0.2
65. docker exec b1 ping -c 4 10.0.0.3
66. docker exec b2 ping -c 4 10.0.0.6
67. docker exec r1 ping -c 4 11.0.0.3
68.
69. docker exec b1 arp -d 10.0.0.4
70. docker exec b1 ping -c 4 10.0.0.4
71.
72. Ubuntu 2 Configuration:
73. sudo ifconfig ens3 20.0.0.2/24
74. sudo ip route add default via 20.0.0.254
75.
76. #Create blue and red VxLAN interfaces
77. ip link add blue type vxlan id 33 group 239.1.1.3 ttl 5 dev ens3 dstport 4789
78. ip link add red type vxlan id 66 group 239.1.1.6 ttl 5 dev ens3 dstport 4789
79.
80. #Activate the VxLAN interfaces
81. ip link set blue up
82. ip link set red up
83.
84. #Assign IP addresses to the VxLAN interfaces
85. ip addr add 10.0.0.102/24 dev blue
86. ip addr add 11.0.0.102/24 dev red
87.
88. #Create two macvlan networks called bluenet and rednet
89. docker network create -d macvlan --subnet=10.0.0.0/24 --gateway=10.0.0.102 -o parent=blue
    bluenet
90. docker network create -d macvlan --subnet=11.0.0.0/24 --gateway=11.0.0.102 -o parent=red
    rednet
91.
92. #Create two alpine containers and attach them to bluenet
93. docker run --privileged --net=bluenet --name=b3 --ip=10.0.0.3 -itd alpine /bin/sh
94. docker run --privileged --net=bluenet --name=b4 --ip=10.0.0.4 -itd alpine /bin/sh
95.
96. #Create two alpine containers and attach them to rednet
97. docker run --privileged --net=rednet --name=r3 --ip=11.0.0.3 -itd alpine /bin/sh
98. docker run --privileged --net=rednet --name=r4 --ip=11.0.0.4 -itd alpine /bin/sh
99.
100. docker container ls
101.
102. Ubuntu 3 Configuration:
103. sudo ifconfig ens3 21.0.0.1/24
104. sudo ip route add default via 21.0.0.254
105.
106. #Docker Installation
107. apt-get update
108. apt install docker.io
109. reboot
110. systemctl status docker
111. ip link
112.
113. #Create blue and red VxLAN interfaces
114. ip link add blue type vxlan id 33 group 239.1.1.3 ttl 5 dev ens3 dstport 4789
115.
116. #Activate the VxLAN interfaces
117. ip link set blue up
118.
119. #Assign IP addresses to the VxLAN interfaces
120. ip addr add 10.0.0.103/24 dev blue
121.
122. #Create two macvlan networks called bluenet and rednet

```

```

123. docker network create -d macvlan --subnet=10.0.0.0/24 --gateway=10.0.0.103 -o parent=blue
    bluenet
124.
125. #Create two alpine containers and attach them to bluenet
126. docker run --privileged --net=bluenet --name=b5 --ip=10.0.0.5 -itd alpine /bin/sh
127. docker run --privileged --net=bluenet --name=b6 --ip=10.0.0.6 -itd alpine /bin/sh
128.
129. docker container ls

```

## Docker Swarm without data encryption

```

1. R1 Configuration:
2.
3. conf t
4. int f0/0
5. ip add 20.0.0.254 255.255.255.0
6. ip nat inside
7. no shut
8.
9. conf t
10. int f1/0
11. ip add 21.0.0.254 255.255.255.0
12. ip nat inside
13. no shut
14.
15. int f0/1
16. ip add dhcp
17. ip nat outside
18. no shut
19. exit
20. ip nat inside source list 1 interface FastEthernet0/1 overload
21. end
22.
23. conf t
24. access-list 1 permit 20.0.0.0 0.0.0.255
25. access-list 1 permit 21.0.0.0 0.0.0.255
26. end
27.
28. Ubuntu 1 Configuration:(Master)
29. nano /etc/netplan/50-cloud-init.yaml
30.
31. network:
32.   version: 2
33.   renderer: networkd
34.   ethernets:
35.     ens3:
36.       addresses:
37.         - 20.0.0.1/24
38.       gateway4: 20.0.0.254
39.       nameservers:
40.         addresses:
41.           - 8.8.8.8
42. netplan apply
43. reboot
44.
45. #Master advertise
46. docker swarm init --advertise-addr=20.0.0.1
47. docker node ls
48.
49. #Create two overlay networks named blue and red
50. docker network create --driver=overlay --attachable blue
51. docker network create --driver=overlay --attachable red

```



```
52.
53. #Create containers
54. docker run --privileged --net=blue --name=b1 -itd alpine /bin/sh
55. docker run --privileged --net=blue --name=b2 -itd alpine /bin/sh
56.
57. docker run --privileged --net=red --name=r1 -itd alpine /bin/sh
58. docker run --privileged --net=red --name=r2 -itd alpine /bin/sh
59.
60. #Test
61. docker exec b1 ping -c 4 b3
62. docker exec b1 ping -c 4 b4
63. docker exec b2 ping -c 4 b5
64. docker exec r1 ping -c 4 r4
65.
66. docker exec b1 arp -d 10.0.1.9
67.
68. #Identify containers
69. docker inspect blue
70. docker inspect red
71.
72. Ubuntu 2 Configuration:
73. nano /etc/netplan/50-cloud-init.yaml
74.
75. network:
76.   version: 2
77.   renderer: networkd
78.   ethernets:
79.     ens3:
80.       addresses:
81.         - 20.0.0.2/24
82.       gateway4: 20.0.0.254
83.       nameservers:
84.         addresses:
85.           - 8.8.8.8
86. netplan apply
87. reboot
88.
89. #Worker join
90. docker swarm join
91. docker node ls
92.
93. #Create containers
94. docker run --privileged --net=blue --name=b3 -itd alpine /bin/sh
95. docker run --privileged --net=blue --name=b4 -itd alpine /bin/sh
96.
97. docker run --privileged --net=red --name=r3 -itd alpine /bin/sh
98. docker run --privileged --net=red --name=r4 -itd alpine /bin/sh
99.
100. #Identify containers
101. docker inspect blue
102. docker inspect red
103.
104. Ubuntu 3 Configuration:
105. nano /etc/netplan/50-cloud-init.yaml
106.
107. network:
108.   version: 2
109.   renderer: networkd
110.   ethernets:
111.     ens3:
112.       addresses:
113.         - 21.0.0.1/24
114.       gateway4: 21.0.0.254
115.       nameservers:
116.         addresses:
```

```
117.           - 8.8.8.8
118.
119. netplan apply
120. reboot
121.
122. #Worker join
123. docker swarm join
124. docker node ls
125.
126. #Create containers
127. docker run --privileged --net=blue --name=b5 -itd alpine /bin/sh
128. docker run --privileged --net=blue --name=b6 -itd alpine /bin/sh
129.
130. #Identify containers
131. docker inspect blue
132. docker inspect red
133.
```

## Docker Swarm with data encryption

```

1. R1 Configuration:
2.
3. conf t
4. int f0/0
5. ip add 20.0.0.254 255.255.255.0
6. ip nat inside
7. no shut
8.
9. conf t
10. int f1/0
11. ip add 21.0.0.254 255.255.255.0
12. ip nat inside
13. no shut
14.
15. int f0/1
16. ip add dhcp
17. ip nat outside
18. no shut
19. exit
20. ip nat inside source list 1 interface FastEthernet0/1 overload
21. end
22.
23. conf t
24. access-list 1 permit 20.0.0.0 0.0.0.255
25. access-list 1 permit 21.0.0.0 0.0.0.255
26. end
27.
28. Ubuntu 1 Configuration:(Master)
29. nano /etc/netplan/50-cloud-init.yaml
30.
31. network:
32.   version: 2
33.   renderer: networkd
34.   ethernets:
35.     ens3:
36.       addresses:
37.         - 20.0.0.1/24
38.       gateway4: 20.0.0.254
39.       nameservers:
40.         addresses:
41.           - 8.8.8.8
42. netplan apply
43. reboot
44.
45. #Master advertise
46. docker swarm init --advertise-addr=20.0.0.1
47. docker node ls
48.
49. #Create two overlay networks named blue and red
50. docker network create --opt encrypted --driver=overlay --attachable blue
51. docker network create --opt encrypted --driver=overlay --attachable red
52.
53. #Create containers
54. docker run --privileged --net=blue --name=b1 -itd alpine /bin/sh
55. docker run --privileged --net=blue --name=b2 -itd alpine /bin/sh
56.
57. docker run --privileged --net=red --name=r1 -itd alpine /bin/sh
58. docker run --privileged --net=red --name=r2 -itd alpine /bin/sh
59.
60. #Test
61. docker exec b1 ping -c 4 b3

```

```
62. docker exec b1 ping -c 4 b4
63. docker exec b2 ping -c 4 b5
64. docker exec r1 ping -c 4 r4
65.
66. docker exec b1 arp -d 10.0.1.9
67.
68. #Identify containers
69. docker inspect blue
70. docker inspect red
71.
72. Ubuntu 2 Configuration:
73. nano /etc/netplan/50-cloud-init.yaml
74.
75. network:
76.   version: 2
77.   renderer: networkd
78.   ethernets:
79.     ens3:
80.       addresses:
81.         - 20.0.0.2/24
82.       gateway4: 20.0.0.254
83.       nameservers:
84.         addresses:
85.           - 8.8.8.8
86. netplan apply
87. reboot
88.
89. #Worker join
90. docker swarm join
91. docker node ls
92.
93. #Create containers
94. docker run --privileged --net=blue --name=b3 -itd alpine /bin/sh
95. docker run --privileged --net=blue --name=b4 -itd alpine /bin/sh
96.
97. docker run --privileged --net=red --name=r3 -itd alpine /bin/sh
98. docker run --privileged --net=red --name=r4 -itd alpine /bin/sh
99.
100. #Identify containers
101. docker inspect blue
102. docker inspect red
103.
104. Ubuntu 3 Configuration:
105. nano /etc/netplan/50-cloud-init.yaml
106.
107. network:
108.   version: 2
109.   renderer: networkd
110.   ethernets:
111.     ens3:
112.       addresses:
113.         - 21.0.0.1/24
114.       gateway4: 21.0.0.254
115.       nameservers:
116.         addresses:
117.           - 8.8.8.8
118.
119. netplan apply
120. reboot
121.
122. #Worker join
123. docker swarm join
124. docker node ls
125.
126. #Create containers
```

```
127. docker run --privileged --net=blue --name=b5 -itd alpine /bin/sh
128. docker run --privileged --net=blue --name=b6 -itd alpine /bin/sh
129.
130. #Identify containers
131. docker inspect blue
132.
```