NETWORK INFRASTRUCTURE PROJECT

IPsec-secured L2TPv3 tunnel for implementing VLAN trunking Comparison with VXLANs

Author
Viet-Huy HA
Anh-Kiet DUONG

Contents

1 Introduction

The project aims to study the L2TPv3 protocol, RFC 3931, to create level 2 tunnels. It will be used to "trunking" VLANs and share services such as DHCP. Finally, with an "intelligent" configuration, we will limit the use of the tunnel when hosts want to connect to the Internet by allowing them to do so directly from their "Internet side". In this project, we will apply theories and guidelines to solve the problems posed.

2 Network Architecture

Based on the original schematic of this project and inspired by other TP labs, we configured this project as follows:

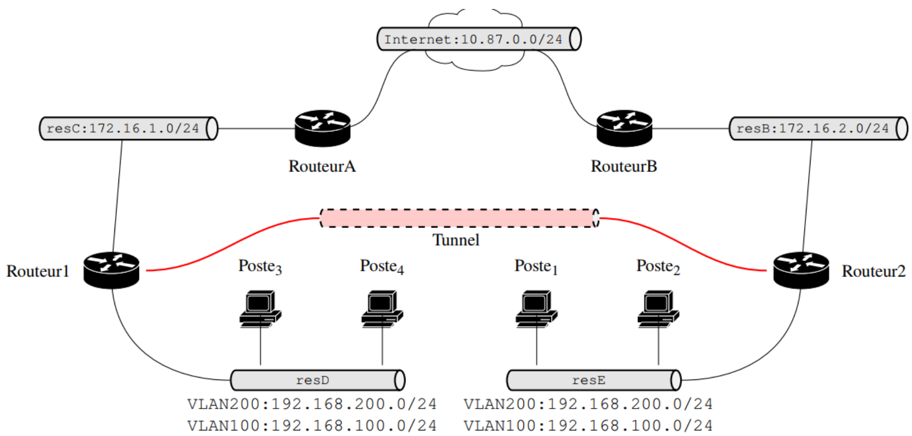


Figure 1: Original Architecture

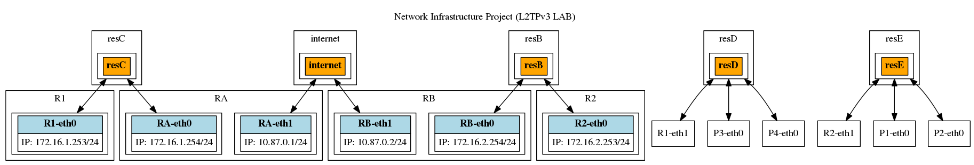


Figure 2: Result Architecture

Note: The initial interface, here are R1-eth1 and R2-eth1 do not have IP configuration.

DHCP Server

We used DHCP servers for assigning IP addresses automatically by using command:

```
ip netns exec R1 dnsmasq -d -z -i R1-eth1.100 --except-interface=lo -F 192.168.100.0,192.168.100.100,255.255.255.0 &
ip netns exec R1 dnsmasq -d -z -i R1-eth1.200 --except-interface=lo -F 192.168.200.0,192.168.200.100,255.255.255.0 &
ip netns exec P3 dhclient P3-eth0.100 &
ip netns exec P4 dhclient P4-eth0.200 &
```

We obtained IP configuration of host P3, P4 and interface R1-eth1:

```
2: P3-eth0.100@P3-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
 qlen 1000
    link/ether 42:0a:1d:b1:01:1c brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.58/24 brd 192.168.100.255 scope global P3-eth0.100
       valid_lft forever preferred_lft forever
    inet6 fe80::400a:1dff:feb1:11c/64 scope link
       valid_lft forever preferred_lft forever
30: P3-eth0@if29: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen
1000
    link/ether 42:0a:1d:b1:01:1c brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::400a:1dff:feb1:11c/64 scope link
       valid_lft forever preferred_lft forever
root@ubuntu:/home/ahn/Desktop/NetInfra/project# [P3]
```

Figure 3: P3 configuration

P3: MAC Address = 42:0a:1d:b1:01:1c, IP Address = 192.168.100.58

```
2: P4-eth0.200@P4-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
 qlen 1000
    link/ether 46:04:8a:2d:77:7d brd ff:ff:ff:ff:ff:ff
    inet 192.168.200.80/24 brd 192.168.200.255 scope global P4-eth0.200
       valid_lft forever preferred_lft forever
    inet6 fe80::4404:8aff:fe2d:777d/64 scope link
       valid_lft forever preferred_lft forever
32: P4-eth0@if31: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen
1000
    link/ether 46:04:8a:2d:77:7d brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::4404:8aff:fe2d:777d/64 scope link
       valid_lft forever preferred_lft forever
root@ubuntu:/home/ahn/Desktop/NetInfra/project# [P4]
```

Figure 4: P4 configuration

P4: MAC Address = 46:04:8a:2d:77:7d, IP Address = 192.168.200.80

```
18: R1-eth0@if17: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 1e:85:0d:07:d4:28 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.16.1.253/24 scope global R1-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::1c85:dff:fe07:d428/64 scope link
       valid_lft forever preferred_lft forever
20: R1-eth1@if19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether ba:ef:9b:f8:89:80 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::b8ef:9bff:fef8:8980/64 scope link
       valid_lft forever preferred_lft forever
root@ubuntu:/home/ahn/Desktop/NetInfra/project# [R1]
```

Figure 5: R1-eth1 configuration

R1-eth1: MAC Address = ba:ef:9b:f8:89:80

3 Problem Solution

3.1 Sniffing network traffic

In this section, we use configuration file named **"Q1_VLAN"**
For this problem, we used "tcpdump" to "sniffing" the network and check if VLAN encapsulation is working:

```
tcpdump -lnvv -i R1-eth1 -e vlan
```

After we use ping command from P3 to P4, we captured traffic on R1-eth1 interface below:

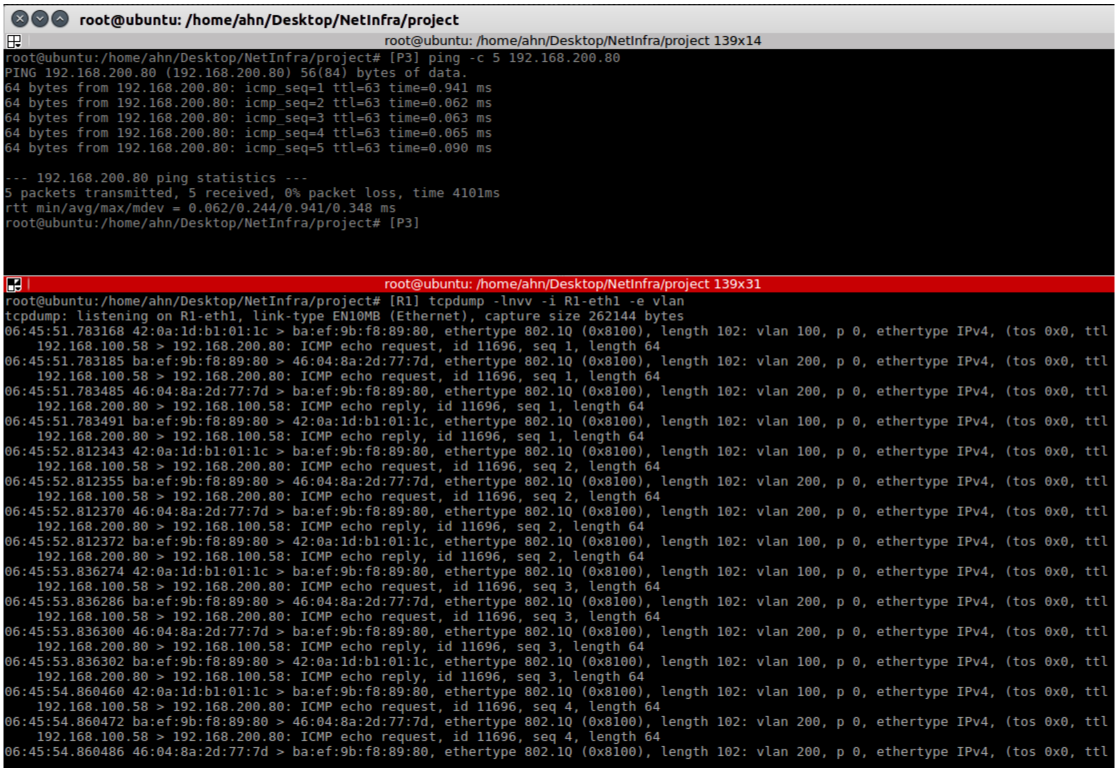


Figure 6: Captured VLAN traffic

Analysis

- As we can see, the first packet is ARP ping between P3 and R1-eth1 interface, VLAN 100
- On next packets, we can see that the router interface R1-eth1 in turn generates an ARP to reach host P4 and transmits the ICMP packet, VLAN 200.

3.2 The L2TPv3 protocol and the VXLANs technology

3.2.1 L2TPv3 and VxLAN Comparison

L2TPv3 (Layer 2 Tunneling Protocol Version 3) and VxLAN (Virtual Extensible Local Area Network, RFC 7348) are both networking technologies designed to create virtual networks and extend Layer 2 connectivity across different locations.

L2TPv3

L2TPv3 is a Layer 2 tunneling protocol that encapsulates Layer 2 frames (such as Ethernet) for transport over IP networks. It is primarily used to interconnect geographically dispersed sites or different VLANs within a data center while maintaining the same Layer 2 domain. L2TPv3 supports 2 packet encapsulation formats: UDP and IP

VxLAN

VxLAN, on the other hand, is a Layer 2 over Layer 3 tunneling protocol that encapsulates Layer 2 Ethernet frames within Layer 3 UDP packets for transport over IP networks. It is specifically designed for data center network virtualization, allowing the creation of large-scale multi-tenant virtual networks and overcoming limitations of traditional VLANs.

- Increased scalability: VLANs are limited to 4096 unique identifiers (based on a 12-bit VLAN ID), which can be insufficient in large-scale, multi-tenant data center environments. VxLAN uses a 24-bit VxLAN Network Identifier (VNI), allowing for up to 16 million unique virtual networks. This significantly increases the number of possible virtual networks, enabling better support for large-scale deployments and multi-tenancy.

- Simplified Layer 2 network design: Traditional VLANs often require complex configurations of Spanning Tree Protocol (STP) to prevent loops and ensure efficient use of links in a Layer 2 network. VxLAN operates at Layer 2 over Layer 3, encapsulating Layer 2 Ethernet frames within Layer 3 UDP packets for transport over IP networks. This approach enables the use of Layer 3 routing and equal-cost multipath (ECMP) load balancing, which simplifies network design and improves network utilization by eliminating the need for STP and its associated complexities.

L2TPv3 is more suited for extending Layer 2 connectivity between sites in a WAN or interconnecting VLANs, while VxLAN is designed for large-scale data center network virtualization. VxLAN offers better scalability and multicast support compared to L2TPv3.

3.2.2 Implementing VxLAN in Linux using Open vSwitch

VxLAN can be implemented in Linux using Open vSwitch (OVS), an open-source software switch designed to work in virtualized environments. OVS supports VxLAN encapsulation, allowing for the creation and management of virtual networks across multiple physical hosts. To configure VxLAN using OVS, you would need to create and configure OVS bridges, add ports, and set the appropriate VxLAN tunnel parameters.
We created a bridge, added port to it and added vxlan1 tunnel on the bridge, using following commands

```
ovs - vsctl add -br br0
ovs - vsctl add - port br0 vxlan1 -- set interface vxlan1 type = vxlan
    options : remote_ip = 172.16.3.233 options :key= flow options : dst_port =
     8472
```

3.2.3 L2TPv3/VXLAN traffic encryption

L2TPv3 doesn't provide built-in encryption. However, It can be combined with IPsec to encrypt the tunnel traffic. This provides data confidentiality, integrity, and authentication between the tunnel endpoints.
VxLAN doesn't offer native encryption either. However, It can be secured using different approaches, such as MACsec (for point-to-point links) or by leveraging encryption solutions provided by the underlying physical network, such as IPsec or SSL/TLS VPNs.

3.2.4 Comparing L2TPv3 and VxLAN with MPLS

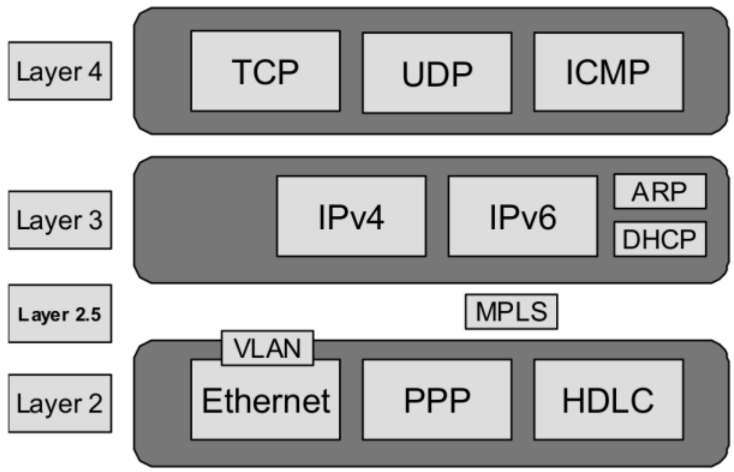


Figure 7: Multiprotocol Label Switching

MPLS (Multiprotocol Label Switching) is a protocol used for high-performance traffic forwarding and network virtualization. It differs from L2TPv3 and VxLAN in that it operates at the Layer 2.5, allowing for more granular control over traffic engineering and Quality of Service (QoS). MPLS can be used to create Layer 2 (VPLS) or Layer 3 (MPLS-VPN) virtual networks.

L2TPv3 and VxLAN rely on IP networks for transport, while MPLS uses label switching for forwarding packets. MPLS provides better traffic engineering capabilities and QoS, but it may require specialized hardware and can be more complex to deploy and manage. On the other hand, L2TPv3 and VxLAN can be deployed using standard IP networks and offer simpler setup and management, particularly in data center environments.

3.3 L2TPv3 tunnel in IP encapsulation mode between R1 and R2

3.3.1 ARP Protocol

In this section, we use configuration file named **Q2_IPTunnel**.
To verify that ARP protocol works normally, we sniff our network by using tcpdump on the tunnel interface on R1 and use a "ping" command sent from P1 to P4.

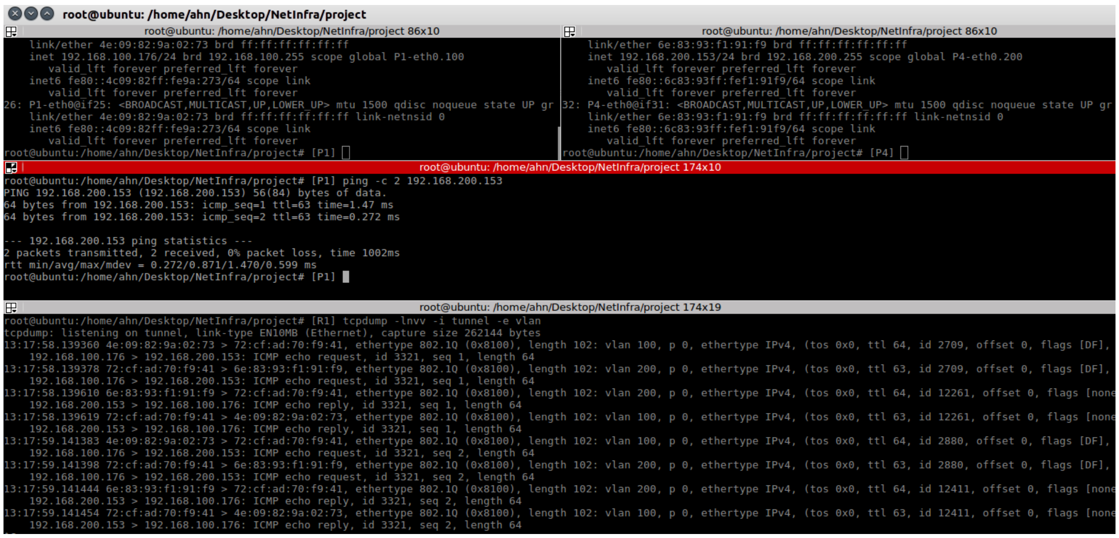


Figure 8: Verify ARP protocol

3.3.2 DHCP Service

Similar to above section, we used DHCP servers for assigning IP addresses automatically by using command:

```
sudo ip netns exec R1 dnsmasq -d -z -i tunnel.100 --except-interface=lo -F
    192.168.100.0,192.168.100.180,255.255.255.0 &
sudo ip netns exec R1 dnsmasq -d -z -i tunnel.200 --except-interface=lo -F
    192.168.200.0,192.168.200.180,255.255.255.0 &
sudo ip netns exec P1 dhclient P1-eth0.100 &
sudo ip netns exec P3 dhclient P3-eth0.100 &
sudo ip netns exec P4 dhclient P4-eth0.200 &
sudo ip netns exec P2 dhclient P2-eth0.200 &
```

Then we got following result:

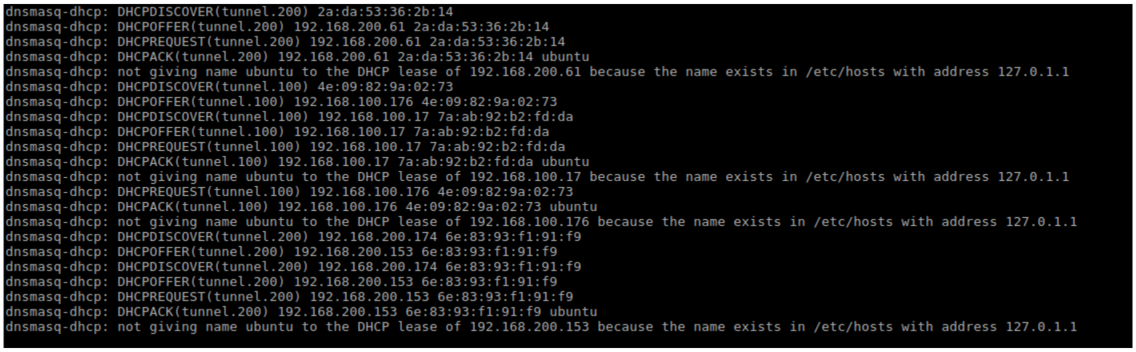


Figure 9: DHCP Server Result

3.3.3 TCP connection

We observed TCP communication between R1 and P1 through tunnel:

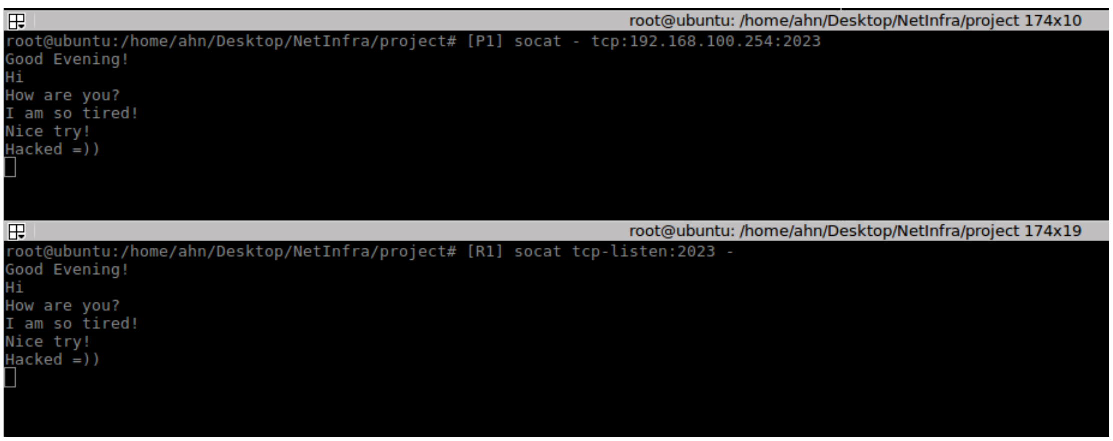


Figure 10: TCP connection

3.4 L2TPv3 IP/UDP Tunnel vs GRE Tunnel Comparison

In this part, we'll use Wireshark for sniffing packets, then analysis to find out difference between encapsulation modes. We capture packets exchange between host P2 and P3.

3.4.1 IP encapsulation mode

Configuration file: Q2_IPTunnel
Captured file: Q4_IP_Communication.pcap.pcapng
To observe the L2TPv3 tunneling in IP mode, we sniff traffic on router RA.
As we can see, L2TPv3 in IP encapsulation mode is displayed by protocol identifier **0x0000** and marked **Unknown** on "tcpdump" console.

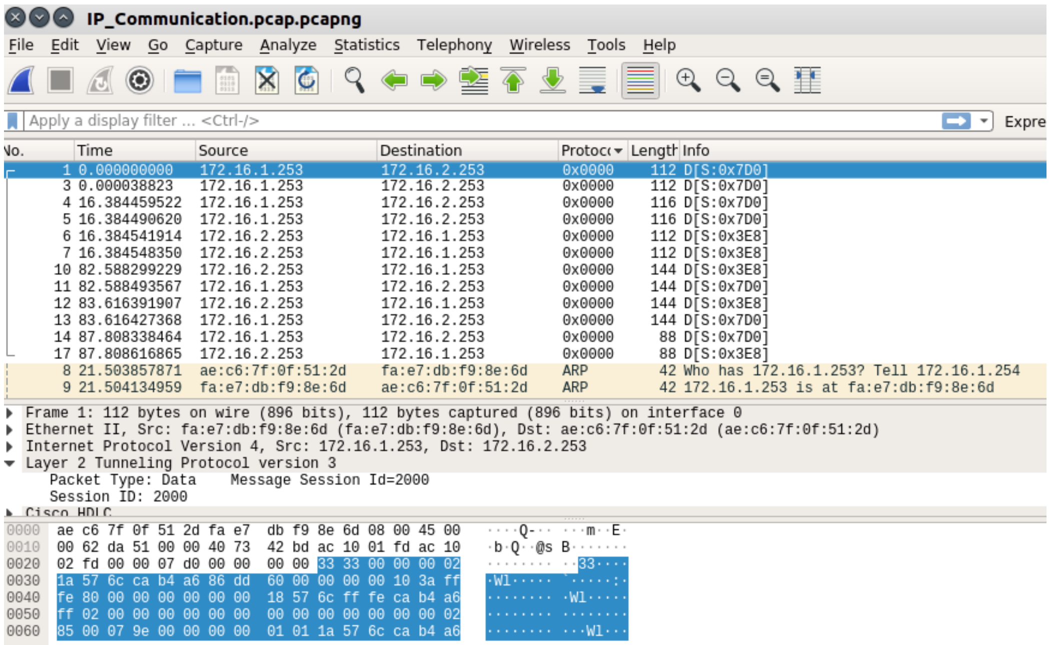


Figure 11: IP encapsulation mode

3.4.2 UDP encapsulation mode

Configuration file: Q4_UDPTunnel

Captured file: Q4_UDP_communication.pcapng

Next, we will monitor traffic of the L2TPv3 tunneling in UDP mode by implementing on port 2022 and 2023.

Through this section, we can clearly see the difference between IP encapsulation mode and UDP encapsulation mode

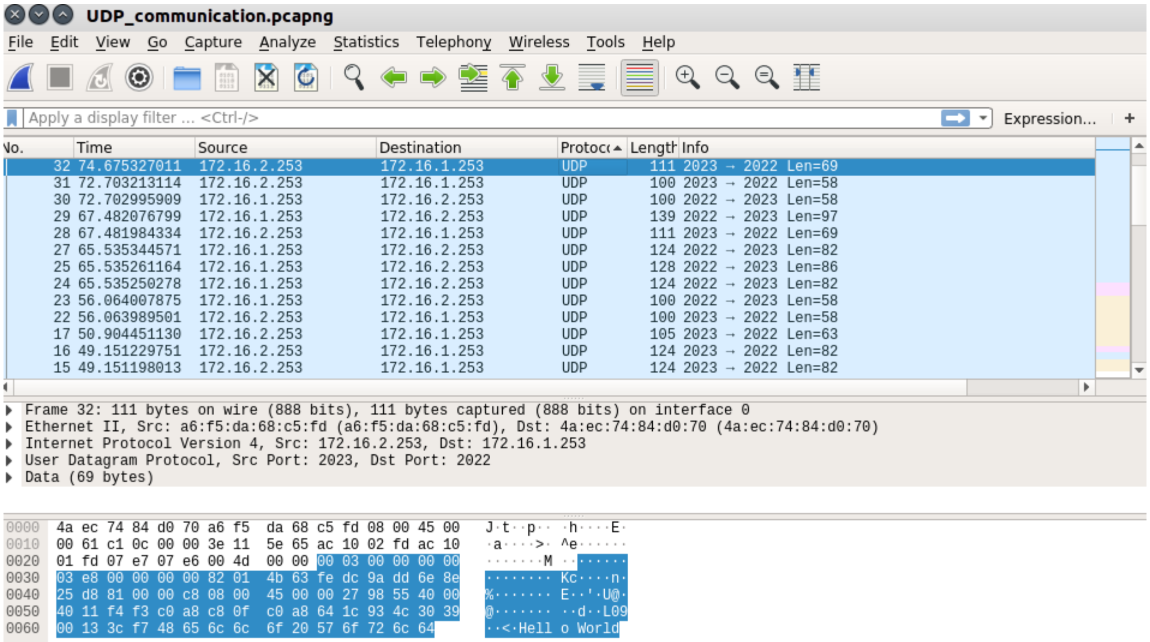


Figure 12: UDP encapsulation mode

3.4.3 GRE

Configuration file: Q4_GRETunnel
Captured file: Q4_GRE_communication.pcapng

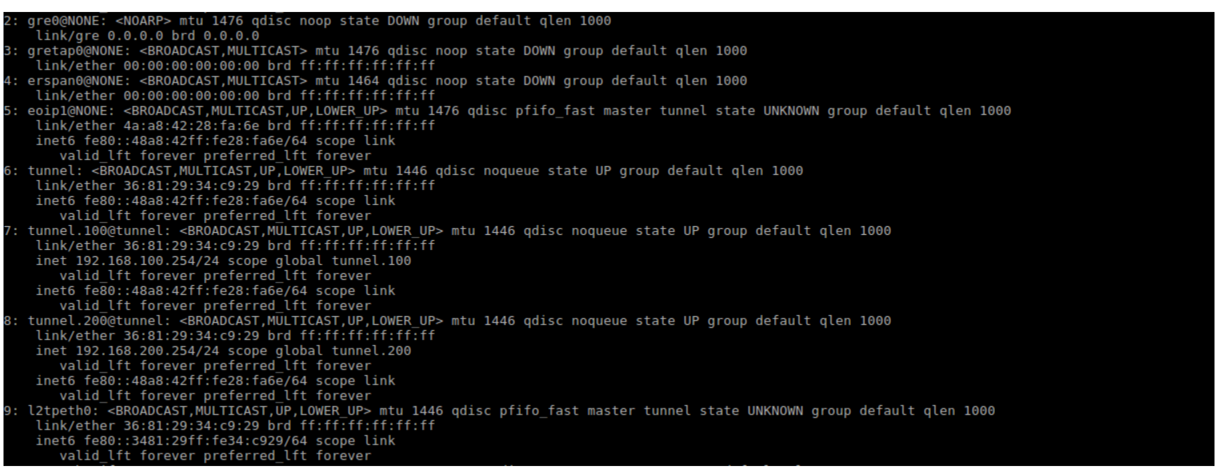


Figure 13: MTU

GRE (Generic Routing Encapsulation) is a tunneling protocol that enables encapsulation of a wide variety of network layer protocols inside point-to-point links. It was developed by Cisco Systems and described in RFC 2784. GRE is commonly used to establish secure and private communication channels between networks over the public internet or other untrusted networks.

Figure 13 shows us the difference between GRE MTU and L2TPv3 MTU. GRE has an MTU length of 1476, while L2TPv3 is only 1446. And we know that GRE header has 24 bytes and L2TPv3 header has 40 bytes.
After configuring GRE in GRETAP mode, we capture following result:

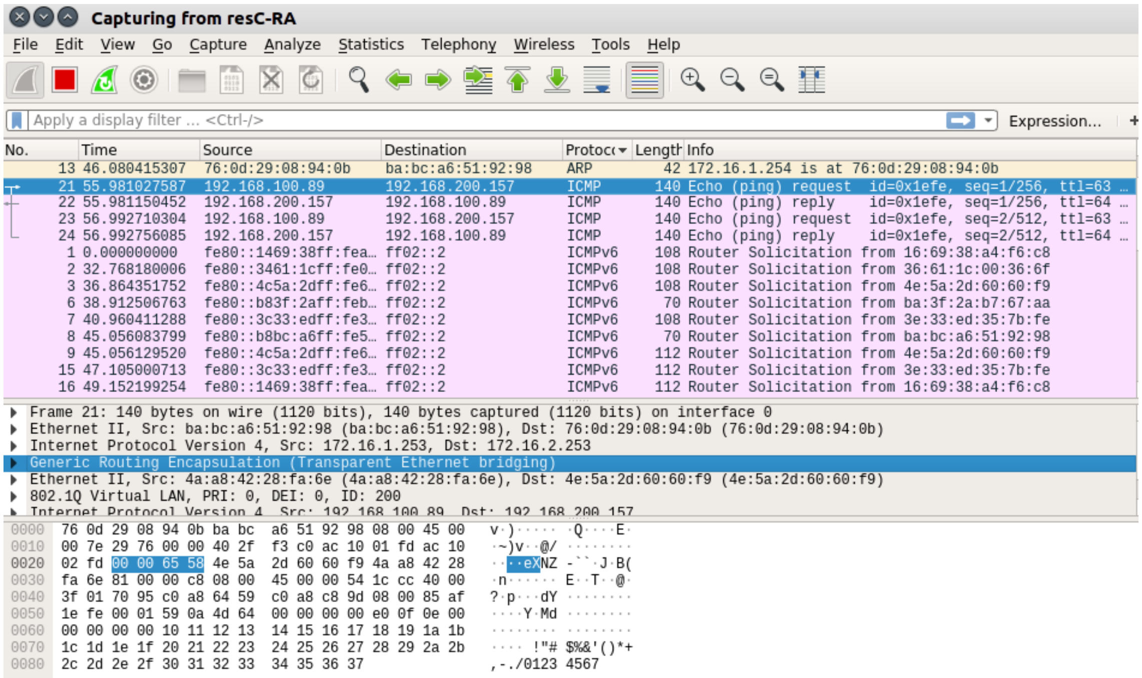


Figure 14: GRETAP

3.5 IPsec on L2TPv3 tunnel

We set up IPsec encryption on our L2TPv3 tunnel by IP encapsulation base on TP files (Configuration can be found in file: **Q5_IPSec**). We will use following command for each router R1 and R2:

```
ip xfrm state flush
ip xfrm policy flush
ip xfrm state add src 172.16.1.253 dst 172.16.2.253 proto esp spi 0x12345678 reqid 0x12345678 mode transport auth sha256 0x323730ed6f1b9ff0cb084af15b197e862b7c18424a7cdfb74cd385ae23bc4f17 enc "rfc3686(ctr(aes))" 0x27b90b8aec1ee32a8150a664e8faac761e2d305b
ip xfrm state add src 172.16.2.253 dst 172.16.1.253 proto esp spi 0x12345678 reqid 0x12345678 mode transport auth sha256 0x44d65c50b7581fd3c8169cf1fa0ebb24e0d55755b1dc43a98b539bb144f2067f enc "rfc3686(ctr(aes))" 0x9df7983cb7c7eb2af01d88d36e462b5f01d10bc1
ip xfrm policy add src 172.16.1.253 dst 172.16.2.253 dir in tmpl src 172.16.1.253 dst 172.16.2.253 proto esp reqid 0x12345678 mode transport
ip xfrm policy add src 172.16.2.253 dst 172.16.1.253 dir out tmpl src 172.16.2.253 dst 172.16.1.253 proto esp reqid 0x12345678 mode transport
```

Next, we capture packets pass through through tunnel to observe the effect of IPsec. Continue to use Wireshark, we got:

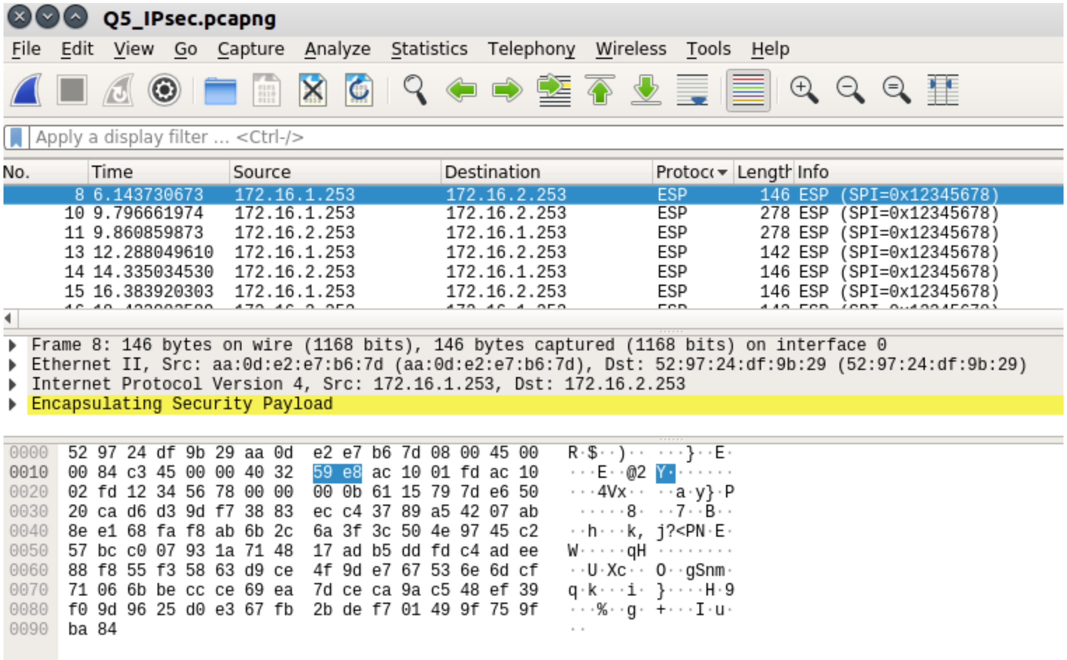


Figure 15: IPsec packet

As shown, there is only a header named Encapsulating Security Payload (ESP) and the protocol of packet is also ESP. There is no information about the IP header nor the data field.
Finally, we will look at transfer speed between router R1 and host P1, then make a comparison about "Transferring without IPsec" and "Transferring with IPsec". The following table is the result obtained when using the **iperf** command:

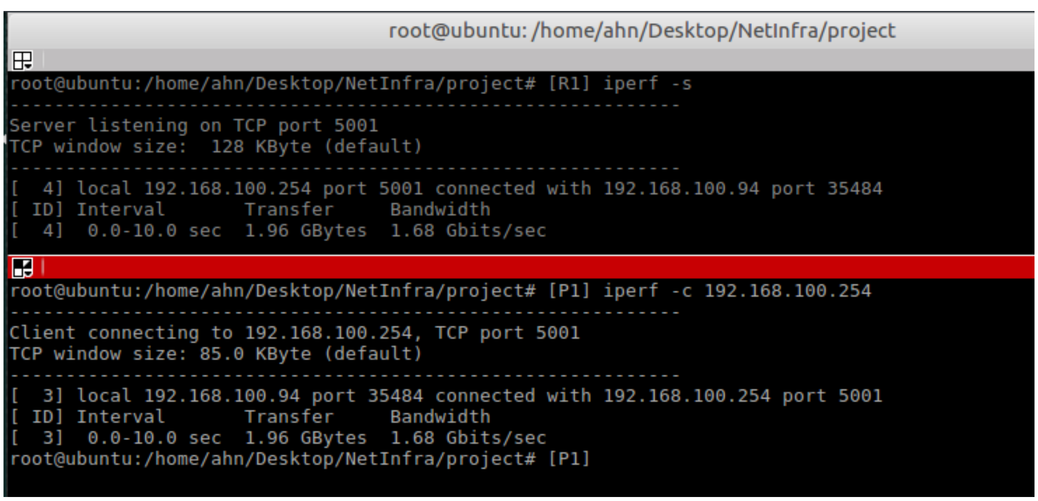
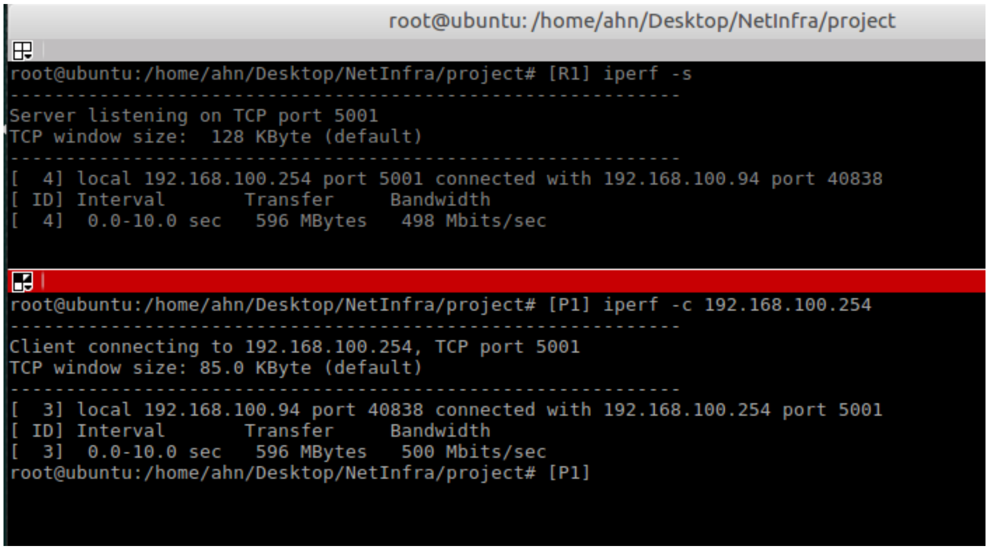


Figure 16: Transferring without IPsec



```
root@ubuntu: /home/ahn/Desktop/NetInfra/project
root@ubuntu:/home/ahn/Desktop/NetInfra/project# [R1] iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  4] local 192.168.100.254 port 5001 connected with 192.168.100.94 port 40838
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec   596 MBytes   498 Mbits/sec

root@ubuntu:/home/ahn/Desktop/NetInfra/project# [P1] iperf -c 192.168.100.254
------------------------------------------------------------
Client connecting to 192.168.100.254, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.100.94 port 40838 connected with 192.168.100.254 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   596 MBytes   500 Mbits/sec
root@ubuntu:/home/ahn/Desktop/NetInfra/project# [P1]
```

Figure 17: Transferring with IPsec

A clear victory in speed was demonstrated for an interval of 10s when **iperf** has transfer 1.96 GBytes and 1.68 GBits/sec Bandwidth without IPsec. Meanwhile, there is only 596 MBytes and 500 MBits/sec Bandwidth for transfer with IPsec. This causes of IPsec encapsulated the IP packets. As a result, the IP packet became longer and lead to fragmentation. Therefore, The receiver needs to reassemble packets.

3.6 "Intelligent" Internet access

Configuration file: Q6_InternetAccess

3.6.1 Access to the Internet

To config for Internet Access, we will use following command:

```
sysctl net.ipv4.conf.all.forwarding=1
#   internet
ip a add dev internet 10.87.0.254/24
iptables -t nat -A POSTROUTING -s 10.87.0.0/24 -j MASQUERADE
#   RA
ip netns exec RA ip route add default via 10.87.0.254
ip netns exec RA iptables -t nat -A POSTROUTING -s 172.16.1.0/24 -j MASQUERADE
#   RB
ip netns exec RB ip route add default via 10.87.0.254
ip netns exec RB iptables -t nat -A POSTROUTING -s 172.16.2.0/24 -j MASQUERADE
```

Then we will check connection by using ping command to Google DNS server, which has IPv4 address 8.8.8.8 or 8.8.8.4.

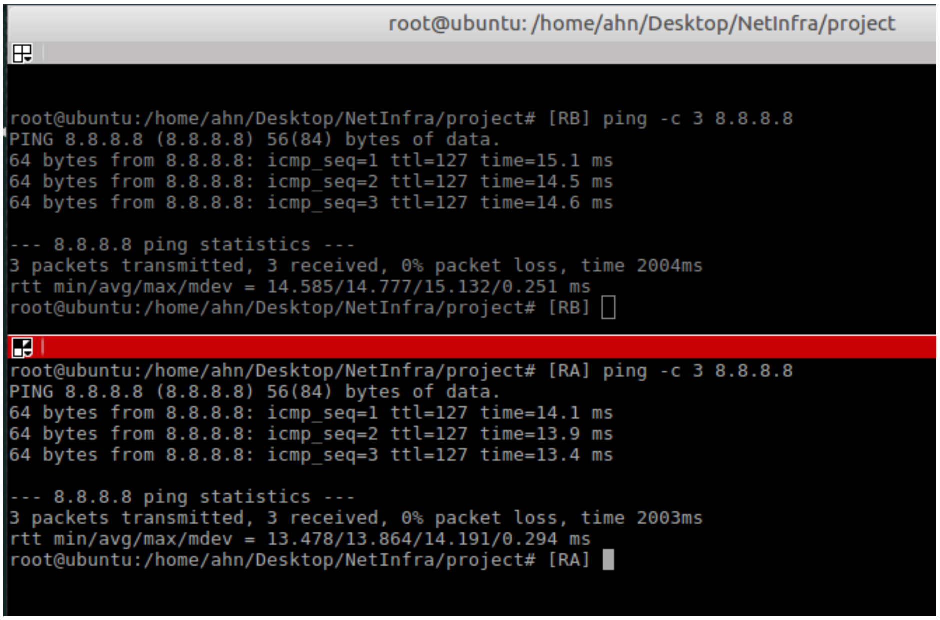


Figure 18: RA and RA access Internet

3.6.2 Access to the Internet on R1 and R2

On routers R1 and R2, for the hosts of the two VLANs can access to the Internet, we use following iptables rules:

```
#   R1
ip netns exec R1 iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -j
    MASQUERADE
ip netns exec R1 iptables -t nat -A POSTROUTING -s 192.168.200.0/24 -j
    MASQUERADE
#   R2
ip netns exec R2 iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -j
    MASQUERADE
ip netns exec R2 iptables -t nat -A POSTROUTING -s 192.168.200.0/24 -j
    MASQUERADE
```

Then we get the results:

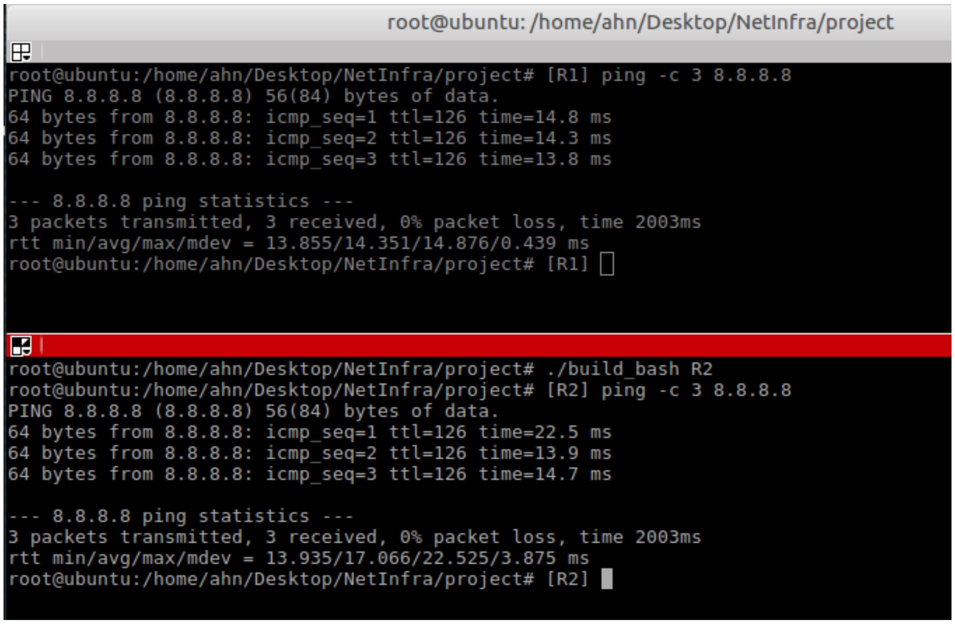


Figure 19: RA and RA access Internet

3.6.3 P1 accesses Internet through R1

To trace path packet taken from host P1 accesses to the Internet, we will use **traceroute** command. Then we can see from P1, packet jumps to next hop R1 (192.168.100.254)

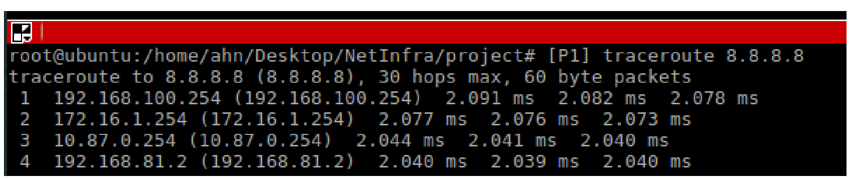


Figure 20: Traffic from P1 goes through R1

3.6.4 Redirect the P1 and P2 towards R2

We will set up a DHCP Relay on R2 for redirecting traffic from P1 and P2 to R2 instead of R1. The DHCP Relay will allows DHCP requests from the hosts on its side to be relayed to the DHCP server located on R1 while specifying the default route to use is an interface to itself. We use following command:

```
sudo ip netns exec R2 dnsmasq -d --dhcp-relay
    =192.168.100.253,192.168.100.254,tunnel.100 --dhcp-relay
    =192.168.100.253,192.168.200.254,tunnel.200 --dhcp-option=tunnel
    ,3,192.168.100.253
```

3.7 Prohibit traffic between VLAN100 and VLAN200

3.7.1 Using "iptables" rules

Configuration file: Q7_ProhibitTraffic_iptables We will use following command:

```
# R1
ip netns exec R1 iptables -t filter -A FORWARD -s 192.168.200.0/24 -d
    192.168.100.0/24  -j DROP
ip netns exec R1 iptables -t filter -A FORWARD -s 192.168.100.0/24 -d
    192.168.200.0/24  -j DROP
# R2
ip netns exec R2 iptables -t filter -A FORWARD -s 192.168.200.0/24 -d
    192.168.100.0/24  -j DROP
ip netns exec R2 iptables -t filter -A FORWARD -s 192.168.100.0/24 -d
    192.168.200.0/24  -j DROP
```

With these commands, all traffic with the source IP address of VLAN100 and the destination IP address of VLAN200 and vice versa is dropped.

3.7.2 Using "Policy Routing"

Configuration file: Q7_ProhibitTraffic_policyrouting
For the prohibition with Policy routing, we added two new routing tables vlan100 and vlan200 into file /etc/iproute2/rt_tables

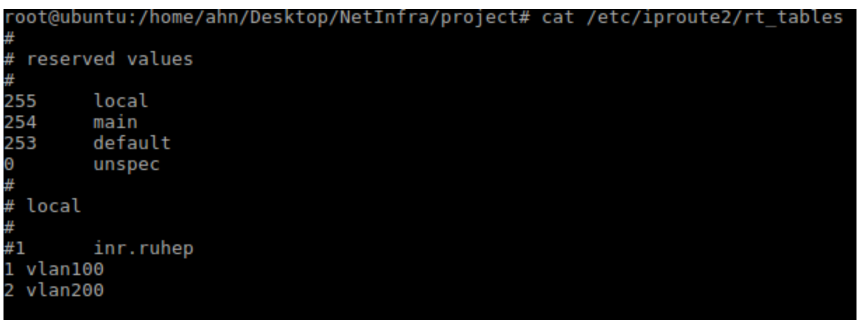


Figure 21: /etc/iproute2/rt_tables

Then we execute following command:

```
# R1
ip netns exec R1 ip rule add from 192.168.100.0/24 lookup vlan100
ip netns exec R1 ip rule add from 192.168.200.0/24 lookup vlan200
ip netns exec R1 ip route add prohibit 192.168.200.0/24 table vlan100
ip netns exec R1 ip route add prohibit 192.168.100.0/24 table vlan200
# R2
ip netns exec R2 ip rule add from 192.168.100.0/24 lookup vlan100
ip netns exec R2 ip rule add from 192.168.200.0/24 lookup vlan200
ip netns exec R2 ip route add prohibit 192.168.200.0/24 table vlan100
ip netns exec R2 ip route add prohibit 192.168.100.0/24 table vlan200
```

3.7.3 Result

After apply above rules or policy, we have final results:

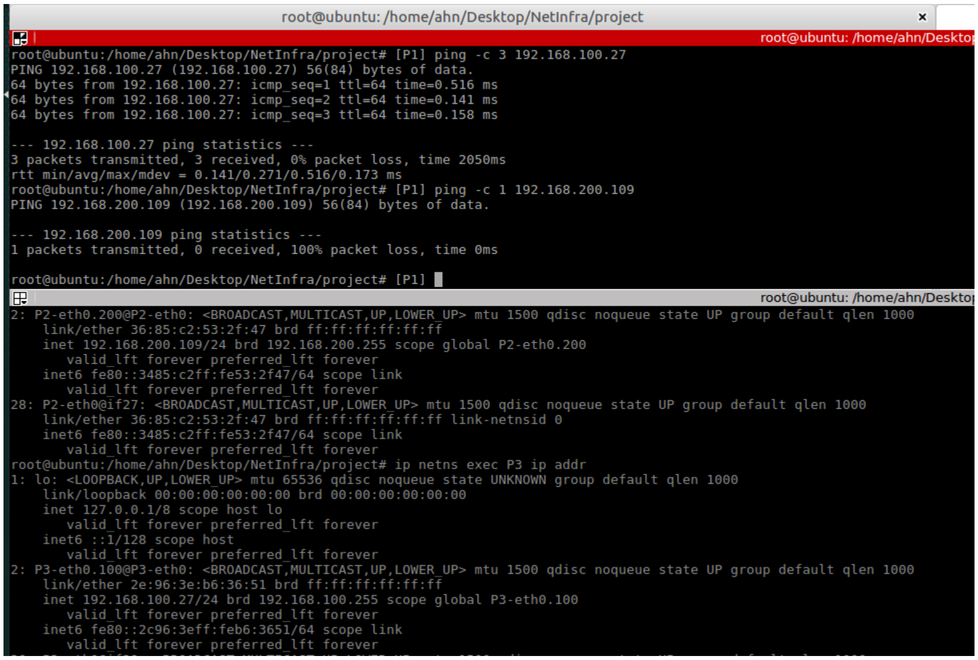


Figure 22:

To verify our configuration, we try to ping from P1 (VLAN100) to P3 (VLAN100) and P2 (VLAN200). We can not ping from P1 to P2 after applying the above configurations.

Appendices

Guide for setting lab

Firstly, Run *build_architecture*

1. Question 1: Run *Q1_VLAN*
2. Question 3: Run *Q2_IPTunnel*
3. Question 4: Run one of three option:
   - *Q2_IPTunnel*
   - *Q4_UDPTunnel*
   - *Q4_GRETunnel*
4. Question 5: Run *Q2_IPTunnel* + *Q5_IPSec*
5. Question 6: Additionally, Run *Q6_InternetAccess*
6. Question 7: Follow report

   Note: After use config for each question, Run *killdns* and *clean*