Report: Infrastructures Reseaux

Nguyen Viet Sang, **Le Tan Nhat Linh**
viet-sang.nguyen@etu.unilim.fr, tan-nhat-linh.le@etu.unilim.fr

Master 1 Cryptis, Faculty of Science and Technique, University of Limoges

1 Introduction

In this project, we study protocol L2TPv3 to implement tunnel of layer 2 and "trunking" VLAN. We compare different mode of protocol L2TP as well as compare L2TP with GRE. For each protocol, we capture the packet and analyse with useful tools such as `tcpdump` and Wireshark. We also configure to use DHCP service and access Internet from VLANs.

This report is organised as follows:

- Section 2 presents the implementation of network architecture.
- Section 3 compares protocols L2TPv3 and VXLANs.
- Section 4 presents the implementation of L2TP tunnel. We also give some discussion about connection between routers and hosts.
- Section 5 compares the implementation of L2TP tunnel with two modes IP and UDP. We also implement GRE tunnel and compare with L2TP tunnel.
- Section 6 shows the implementation of IPsec. The comparison of packets with IPsec with the ones without IPsec is also analysed.
- Section 7 shows the work of configuration to access Internet.
- Section 8 shows the configuration of `iptables` and Policy Routing to forbid traffic from different VLAN.
- Section 9 concludes our work.

2 Implement network architecture

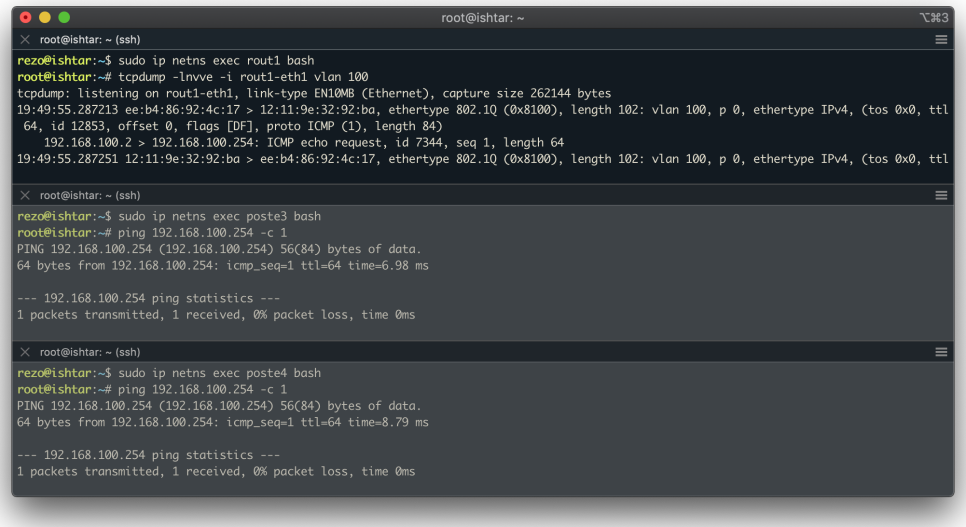


Fig. 1: Ping from `poste3` to `rout1`

To implement VLAN, besides `tunnel.100` and `tunnel.200` on interface `tunnel`, we also add VLAN to each interface of corresponding `poste`. For example, `poste3-eth0.100` is added on interface

poste3-eth0. Therefore, we can ping from poste3 to rout1 (Router 1). Figure 1 shows the result of pinging from poste3 (192.168.100.2) to interface tunnel.100 (192.168.100.254). tcpdump on interface rout1-eth1 only captures packets tagged VLAN100 (listing 1.1). When we ping this interface from poste4, it will not capture packets because these packets are tagged as VLAN200.

```
tcpdump -lnvve -i rout1-eth1 vlan 100
```

Listing 1.1: tcpdump on rout1-eth1

We analyse the result in figure 1. 192.168.100.2 > 192.168.100.254 means packet is sent from poste3 to rout1. MAC address of poste3 is ee:b4:86:92:4c:17. MAC address of tunnel is 12:11:9e:32:92:ba. The protocol is ICMP. These packets belong to VLAN100. Lengths of packets are 102.

3 Presentation of L2TPv3 and VXLANs

3.1 Compare two solutions

VXLAN: In the case when the VMs in a data center are grouped according to their Virtual LAN (VLAN), one might need thousands of VLANs to partition the traffic according to the specific group to which the VM may belong. The current VLAN limit of 4094 is inadequate in such situations. One more common problem is that each user may independently assign MAC addresses and VLAN IDs leading to potential duplication of these on the physical network [5]. VXLAN is a tunneling protocol designed to solve these problems.

In short, VXLAN is a Layer 2 overlay scheme on a Layer 3 network. This overlay is used to carry the MAC traffic from the individual VMs in an encapsulated format over a logical "tunnel". Each overlay is termed a VXLAN segment. Only VMs within the same VXLAN segment can communicate with each other. Each VXLAN segment is identified through a 24-bit segment ID, termed the "VXLAN Network Identifier (VNI)". This allows up to 16M VXLAN segments to coexist within the same administrative domain [5].

Unlike most tunnels, a VXLAN is a 1 to N network, not just point to point. A VXLAN device can learn the IP address of the other endpoint either dynamically in a manner similar to a learning bridge, or make use of statically-configured forwarding entries.

L2TPv3: Layer 2 Tunnelling Protocol Version 3 is used as an alternative protocol to MPLS for encapsulation of multi-protocol Layer 2 communications traffic over IP networks. Like L2TP, L2TPv3 provides a pseudo-wire service, but scaled to fit carrier requirements.

L2TP extends the Point-to-Point Protocol (PPP) model by allowing the L2 and PPP endpoints to reside on different devices interconnected by a packet-switched network [3]. L2TP provides a dynamic mechanism for tunneling Layer 2 (L2) "circuits" across a packet-oriented data network. L2TPv3 has increase the number of bits of Session ID and Tunnel ID from 16 to 32. This protocol aslo extend the Tunnel Authentication mechanism to cover the entire control message rather than just a portion of certain messages [4].

3.2 Implement VXLAN on Linux

The commands in listing 1.2 create a new interface acting as a VXLAN tunnel endpoint, named vxlan100 and put it in a bridge with some regular interfaces.

```
ip link add vxlan100 type vxlan dev rout1-eth1 id 42 local 172.16.1.253
    remote 172.16.2.253 dstport 4789
brctl addbr br100
brctl addif br100 vxlan100
brctl addif br100 rout1-eth0
ip link set up dev br100
ip link set up dev vxlan100
```

Listing 1.2: Implementation of VXLAN on Linux

3.3 Encrypt L2TPv3 and VXLAN traffic

To encrypt L2TPv3 or VXLAN traffic, we can use MACsec [1] which is an IEEE standard for security in wired ethernet LANs. MACsec is a Layer 2 protocol that relies on GCM-AES-128 to offer integrity and confidentiality, and operates over ethernet. It can secure all traffic within a LAN, including DHCP and ARP, as well as traffic from higher layer protocols. It is an extension to 802.1X provides secure key exchange and mutual authentication for MACsec nodes.

MACsec is also compatible with many tunneling technologies such as VXLAN and L2TPv3. A cloud customer with a virtual private LAN can use MACsec to encrypt all the internal traffic before it leaves the virtual machines. That way the cloud provider cannot peek into the communication between the VMs.

3.4 Compare with MPLS

Both MPLS and VXLAN require specific hardware support to operate at line rate but VXLAN only requires hardware support for encapsulation at the edge of the network and thus network cores do not necessarily need replacing. MPLS works in the network cores and demands end to end support [2]. L2TPv3 is also works at the edge of the network.

4 Implement L2TP tunnel

To establish L2TP tunnel, we must connect `rout1` and `rout2`. We connect these two routers by static routes as listing 1.3 below. Then, we can establish L2TP tunnel between `rout1` and `rout2` as the provided instruction. For routing on each router, we also can use RIP or OSPF as presented in TP1 and TP2. However, in this project, we use static configuration as shown in listing 1.3.

```
ip netns exec rout1 ip r add 10.87.0.0/24 via 172.16.1.254
ip netns exec rout1 ip r add 172.16.2.0/24 via 172.16.1.254
ip netns exec rout2 ip r add 10.87.0.0/24 via 172.16.2.254
ip netns exec rout2 ip r add 172.16.1.0/24 via 172.16.2.254
ip netns exec routA ip r add 172.16.2.0/24 via 10.87.0.2
ip netns exec routB ip r add 172.16.1.0/24 via 10.87.0.1
```

Listing 1.3: Connect `rout1` and `rout2`

After implement L2TP tunnel, we verify protocol ARP which discovers different machines on the both sides by run `arp` as shown in figure 2. We can see that protocol ARP normally operates and matches MAC addesses of `poste1`, `poste2`, `poste3` and `poste4` to their IP addresses (192.168.100.1, 192.168.200.1, 192.168.100.2, 192.168.200.2, respectively). It is the same for the other side.

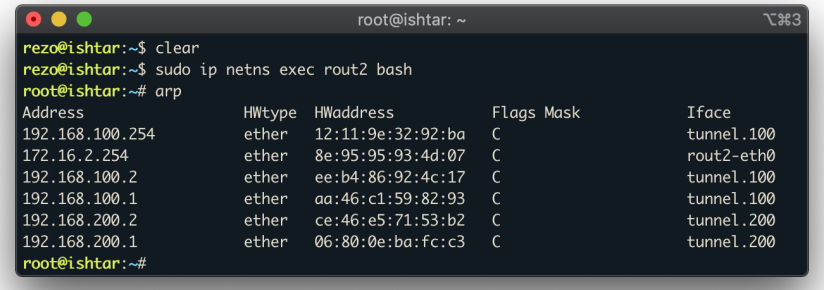


Fig. 2: Verify protocol ARP

We verify DHCP service on `rout1` as shown in figure 3. This service provides an IP range 192.168.100.5-192.168.100.9 on interface `tunnel.100` by the command in listing 1.4. When we run

`dhclient poste1-eth0.100` on `poste1`, it can receive IP address (192.168.100.5) in the range above. We do the same with other `postes` and other VLANs.

```
dnsmasq -d -z -i tunnel.100 -F 192.168.100.5,192.168.100.9,255.255.255.0
```

Listing 1.4: Run DHCP service on `rout1`

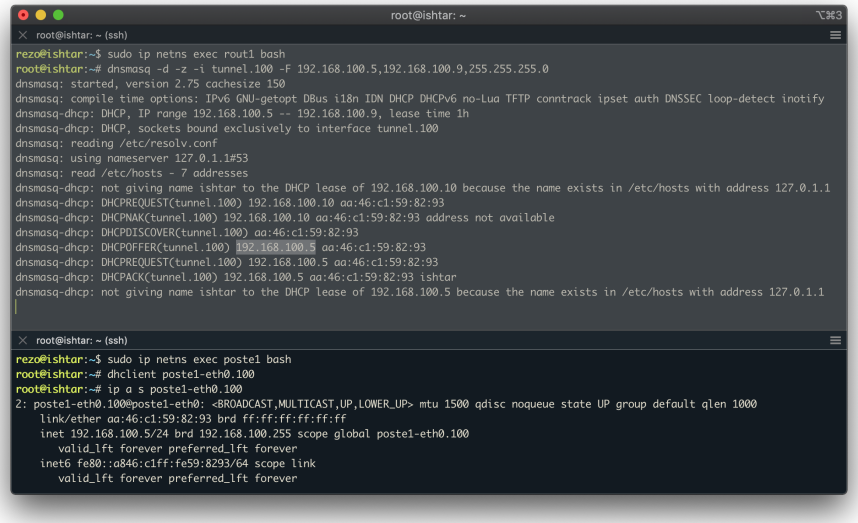


Fig. 3: DHCP service on `rout1`

We now establish a TCP connection between `rout1` and `poste1` through tunnel as shown in figure 4. As we can see, we can exchange message between `poste1` and `rout1` (interface `tunnel.100`, 192.168.100.254) through the tunnel.

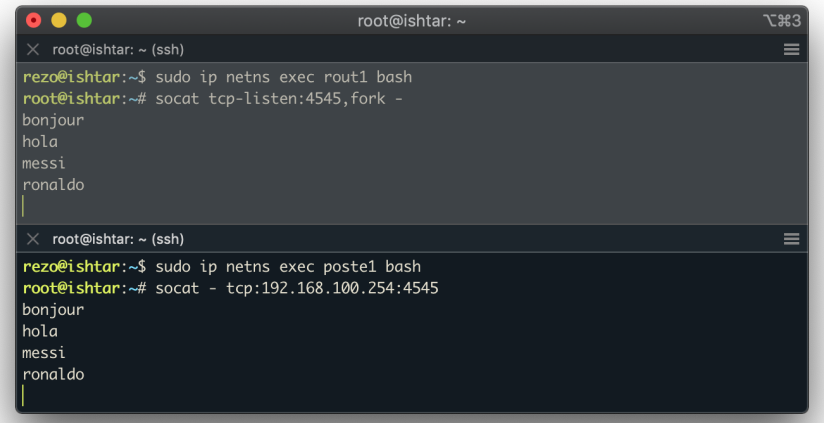


Fig. 4: TCP connection between `poste1` and `rout1` thanks to `socat`

5 Encapsulation modes IP and UDP. Tunnel GRE with GRETAP

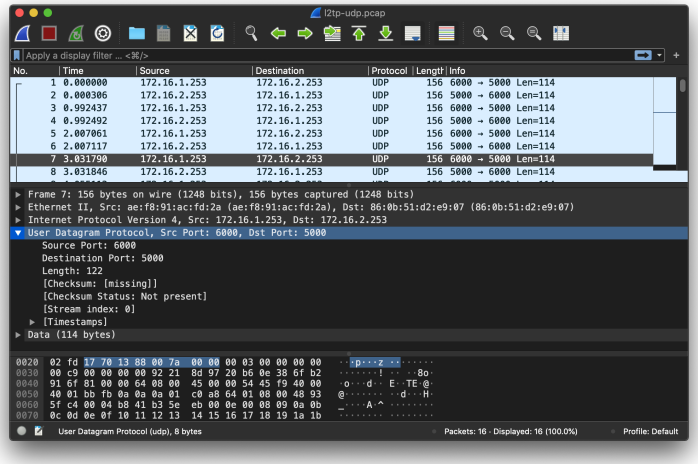
5.1 Implementation of L2TPv3 with modes IP and UDP

To implement L2TP tunnel with mode UDP, it is necessary to declare ports of connection on tunnel as shown in listing 1.5

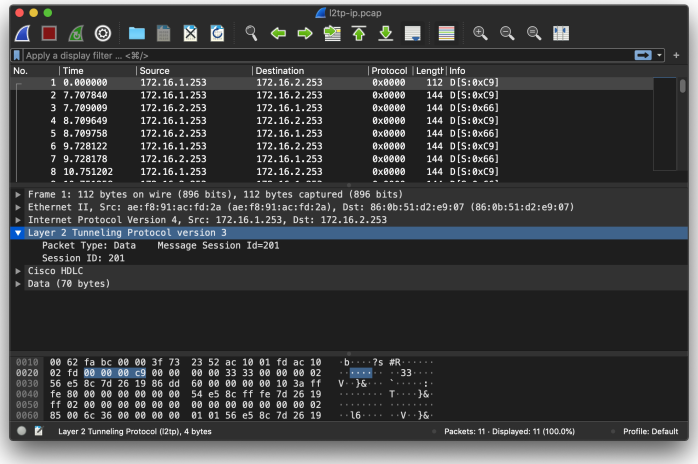
```
ip netns exec rout1 ip l2tp add tunnel tunnel_id 12 peer_tunnel_id 21 encap
    udp local 172.16.1.253 remote 172.16.2.253 udp_sport 6000 udp_dport 5000
ip netns exec rout2 ip l2tp add tunnel tunnel_id 21 peer_tunnel_id 12 encap
    udp local 172.16.2.253 remote 172.16.1.253 udp_sport 5000 udp_dport 6000

ip netns exec rout2 ip l2tp add tunnel tunnel_id 21 peer_tunnel_id 12 encap
    udp local 172.16.2.253 remote 172.16.1.253 udp_sport 5000 udp_dport 6000
ip netns exec rout2 ip l2tp add session tunnel_id 21 session_id 201
    peer_session_id 102
# ... the same configuration with bridge and l2tpeth0 as mode IP
```

Listing 1.5: L2TP tunnel with mode UDP



(a) L2TP tunnel mode UDP



(b) L2TP tunnel mode IP

Fig. 5: Packets with L2TP header

To see the difference, we capture packets through tunnel by listening on `routA` or `routB` thanks to `tcpdump`. We then use Wireshark to explore inside the packets as show in figure 5. As we can see, although we ping with the IP addresses 192.168.100.254 and 192.168.200.254 (belong to network of tunnel), the source IP and destination IP in these packets are 172.16.1.253 (`rout1-eth0`) and 172.16.2.253 (`rout2-eth0`).

In figure 5a, when we use L2TP tunnel with mode UDP, packets contain an UDP header with the same source port and destination port as our configuration. In figure 5b, when we use L2TP tunnel with mode IP, packets contain a L2TPv3 header. Besides, the length of packets with mode UDP greater than the one with mode IP (156 compared to 144) because the difference length of these header. Moreover, protocols of packets are also different (UDP compared to 0x0000).

5.2 Compare with GRE tunnel mode GRETAP

After creating tunnel GRE with mode GRETAP, we also add VLAN to interfaces `eoipl` (listing 1.6) and ping between `rout1` and `rout2`. Figure 6 shows packets captured through this tunnel. As we can see, there are something difference. Source and destination IP addresses are the addresses we assign for interfaces `eoipl`. Protocol is ICMP. Especially, packets have a GRE header. This protocol type is a transparent ethernet bridging.

```
ip netns exec rout1 ip link add eoipl type gretap local 172.16.1.253 remote
    172.16.2.253
ip netns exec rout1 ip link set dev eoipl up
ip netns exec rout2 ip link add eoipl type gretap remote 172.16.1.253 local
    172.16.2.253
ip netns exec rout2 ip link set dev eoipl up

ip netns exec rout1 ip link add link eoipl name eoipl.100 type vlan id 100
ip netns exec rout1 ip link set dev eoipl.100 up
ip netns exec rout1 ip link add link eoipl name eoipl.200 type vlan id 200
ip netns exec rout1 ip link set dev eoipl.200 up
ip netns exec rout1 ip a add 10.10.10.2/24 dev eoipl.100
ip netns exec rout1 ip a add 10.10.20.2/24 dev eoipl.200
# ... the same on rout2
```

Listing 1.6: Configuration of GRE tunnel

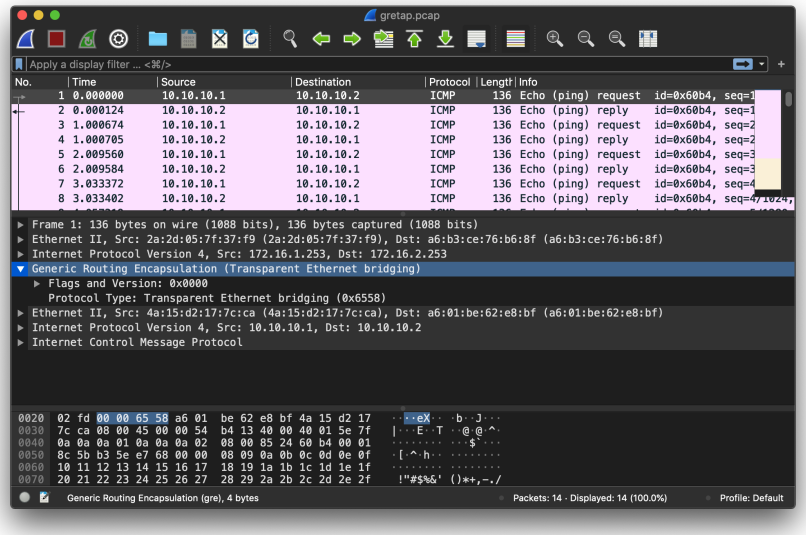


Fig. 6: Packets through GRE tunnel

We also notice that an Ethernet header and an IP header (src:10.10.10.10.1, dst:10.10.10.2) is wrapped in side another Ethernet header and another IP header (src: 172.16.1.253, dst:172.16.2.253). MTU of GRE packets, L2TP mode IP and L2TP mode UDP are 1462, 1492 and 1488, respectively. They are different because of different lengths of headers.

6 Implement IPsec

To implement IPsec encryption on L2TPv3, we configure as listing 1.7.

```
ip netns exec rout1 ip xfrm state flush
ip netns exec rout1 ip xfrm policy flush
ip netns exec rout1 ip xfrm state add src 172.16.1.253 dst 172.16.2.253 \
    proto esp spi 0x12345678 reqid 0x12345678 mode transport auth sha256 \
    0x323730ed6f1b9ff0cb084af15b197e862b7c18424a7cdfb74cd385ae23bc4f1 \
    enc "rfc3686(ctr(aes))" 0x27b90b8aec1ee32a8150a664e8faac761e2d305b
ip netns exec rout1 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 netns exec rout1 ip xfrm policy add src 172.16.2.253 dst 172.16.1.253 \
    dir in tmpl src 172.16.2.253 dst 172.16.1.253 \
    proto esp reqid 0x12345678 mode transport
ip netns exec rout1 ip xfrm policy add src 172.16.1.253 dst 172.16.2.253 \
    dir out tmpl src 172.16.1.253 dst 172.16.2.253 \
    proto esp reqid 0x12345678 mode transport

# The same for rout2, but replace "dir in" (line 12) by "dir out" and "dir
   out" (line 15) with "dir in"
```

Listing 1.7: Configuration of IPsec

We also capture packets through tunnel to see the effect of IPsec (figure 7). As we can see, there is no L2TP header in this packet. Instead, it has a header named `Encapsulating Security Payload` (ESP). The protocol of packet is also ESP. We do not see the field of data in this packet. Length of packet is greater than the one without IPsec (174 compared to 144).

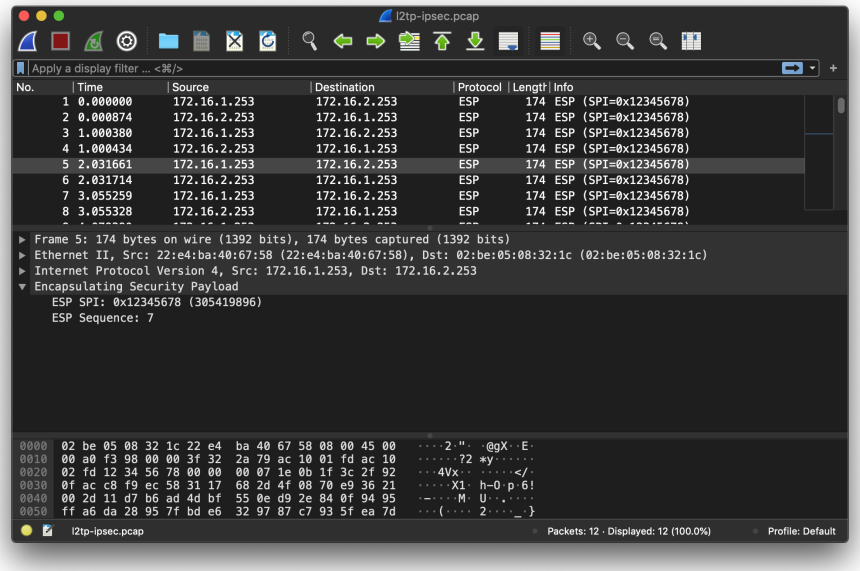
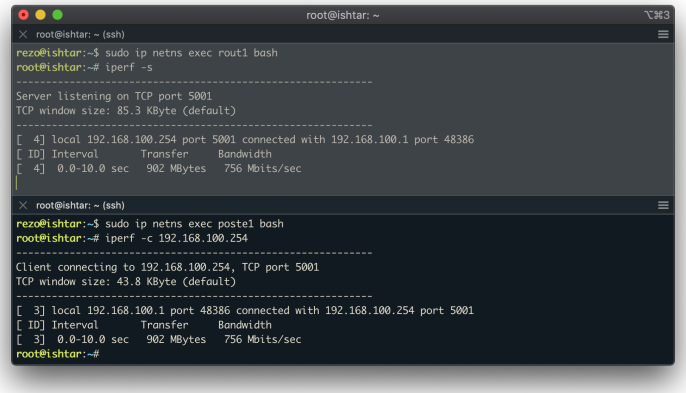
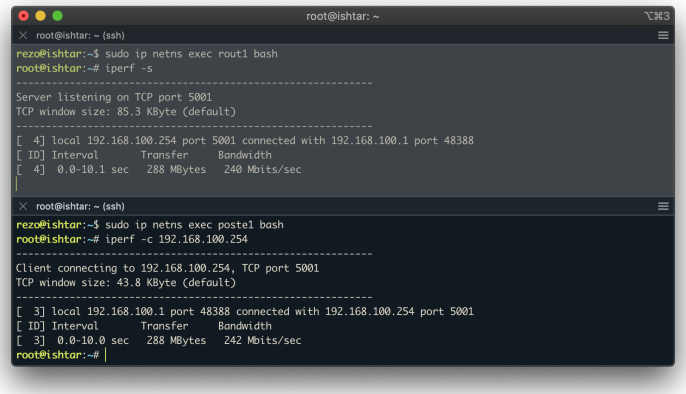


Fig. 7: Packets through tunnel with IPsec on L2TPv3

We use `iperf` to examine the speed of transferring packets between `rout1` and `poste1`. As we can see in figure 8, transferring packets with IPsec is about 3 times slower than without IPsec (240 Mbits/sec in figure 8b compared to 756 Mbits/sec in figure 8a). This happens because features attack defense consume many CPU resources. Besides, IPsec encapsulates IP packets. As a result, the IP packet length becomes longer which leads to fragmentation. The receiver also needs to reassemble packets. Fragmentation, reassembly, encryption and decryption of fragments consume many CPU resources.



(a) Speed of transferring without IPsec



(b) Speed of transferring with IPsec

Fig. 8: Compare speed of transferring

7 Access Internet

To access Internet from switch `internet`, we configure as listing 1.8. We firstly link the switch `internet` with the real machine (VM) (lines 1-6). It needs to define default routes for traffic to pass through and go to the Internet (lines 8-10). It also needs to masquerade traffic from switch `internet` on VM (line 10). With these configuration, we can connect to Internet from switch `internet`. Notice that it should be added `nameserver 8.8.8.8` to the file `/etc/resolv.conf` in order to access Internet with domain names.

```
1 ip link add vm-eth0 type veth peer name internet-vm
2 ovs-vsctl add-port internet internet-vm
3 ip link set dev vm-eth0 up
```

```
ip link set dev internet-vm up
ip a add dev vm-eth0 10.87.0.3/24
sysctl net.ipv4.conf.all.forwarding=1

ip netns exec routA ip r add default via 10.87.0.3
ip netns exec routB ip r add default via 10.87.0.3
iptables -t nat -A POSTROUTING -s 10.87.0.0/24 -j MASQUERADE
```

Listing 1.8: Configure to access Internet

We then configure to be able to access Internet from `rout1` for the VLANs as listing 1.9. We notice that all `postes` are set default routes to interface `tunnel.100` of `rout1` when they use DHCP service on it. Therefore, it is not necessary to configure default routes for each `poste`. `rout1` and `rout2` are set default routes to interfaces `routA-eth1` and `routB-eth1`, respectively. Traffic from switches `resB` and `resC` are masqueraded (lines 8-11). Then, traffic from VLANs also masqueraded on `rout1` and `rout2` (lines 13-19).

```
ip netns exec rout1 ip r add default via 172.16.1.254
ip netns exec rout2 ip r add default via 172.16.2.254

ip netns exec routA iptables -t nat -A POSTROUTING \
    -s 172.16.1.0/24 -j MASQUERADE
ip netns exec routB iptables -t nat -A POSTROUTING \
    -s 172.16.2.0/24 -j MASQUERADE

ip netns exec rout1 iptables -t nat -A POSTROUTING \
    -s 192.168.100.0/24 -j MASQUERADE
ip netns exec rout1 iptables -t nat -A POSTROUTING \
    -s 192.168.200.0/24 -j MASQUERADE
ip netns exec rout2 iptables -t nat -A POSTROUTING \
    -s 192.168.100.0/24 -j MASQUERADE
ip netns exec rout2 iptables -t nat -A POSTROUTING \
    -s 192.168.200.0/24 -j MASQUERADE
```

Listing 1.9: Configure to access Internet for VLANs

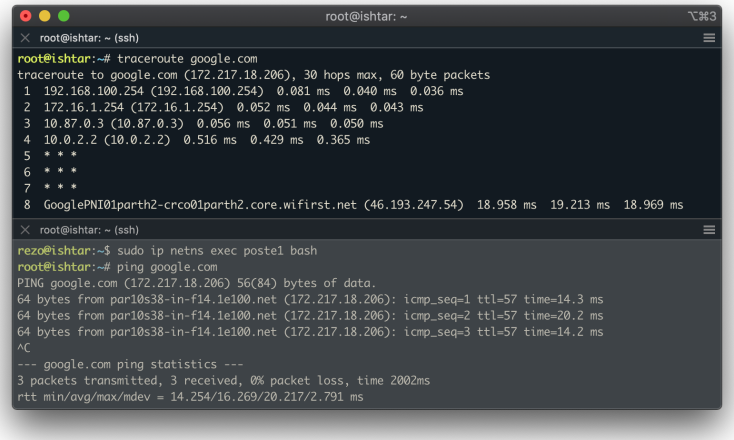


Fig. 9: Access Internet from `poste1`

Figure 9 shows a capture of accessing Internet from `poste1`. We also trace the route and see that, the traffic passes to `rout1` (interface tunnel.100, 192.168.100.254) before going to the Internet.

To redirect traffic from `poste1` (and `poste2`) to the Internet via `rout2` instead of `rout1`, we add one more DHCP service on `rout2`. When having 2 DHCP servers, we would like that `poste1` will get

IP address from DHCP server on `rout2` and hence, `poste1` will have the default route via `rout2`. So, when `poste1` requires an IP address, we specify which DHCP server `poste1` should use as shown in listing 1.10. We notice that the IP ranges of these two DHCP servers are not overlapped.

```
dhclient poste1-eth0.100 -s 192.168.100.253
```

Listing 1.10: Specify which DHCP server to use

Figure 10 shows the result of redirecting traffic via `rout2`. As we can see, there are two DHCP servers running on two routers. When `poste1` get the IP address from DHCP server on `rout2`, it has the default route via `tunnel.100` (192.168.100.253) of `rout2`. As a result, if traffic from `poste1` goes to the Internet, it will pass through `rout2` instead of `rout1`.

Fig. 10: `poste1` accesses Internet via `rout2`

8 Forbid traffic between VLAN100 and VLAN200

To forbid traffic between different VLANs, we configure with `iptables` as listing 1.11. All traffic has soure IP addresses of VLAN100 and destination IP addresses of VLAN200 and vice versa are dropped in table `FORWARD`.

```
ip netns exec rout1 iptables -t filter -A FORWARD \
    -s 192.168.100.0/24 -d 192.168.200.0/24 -j DROP
ip netns exec rout1 iptables -t filter -A FORWARD \
    -s 192.168.200.0/24 -d 192.168.100.0/24 -j DROP
ip netns exec rout2 iptables -t filter -A FORWARD \
    -s 192.168.100.0/24 -d 192.168.200.0/24 -j DROP
ip netns exec rout2 iptables -t filter -A FORWARD \
    -s 192.168.200.0/24 -d 192.168.100.0/24 -j DROP
```

Listing 1.11: Forbid traffic between different VLAN with `iptables`

We also use Policy Routing instead of `iptables` as listing 1.12. If we ping between machines in the same VLAN, it connects directly without being routed by the routers. In contrary, when we ping from a machine of VLAN100 to a machine of VLAN200, for example, from `poste3` to `poste4`, packets must be routed by router `rout1`. At this time, configuration by lines 6-9 in listing 1.12 will prohibit the traffic from VLAN100 to VLAN200. This traffic uses the table 11 of `rout1` to determine its route (line 6). Then, if it has destination of VLAN200, it will be prohibited (lines 7-8). Otherwise, it will go through default route (line 9). The default route is to forward traffic to the Internet.

```
1  ip netns exec rout1 ip r flush 11
2  ip netns exec rout1 ip r flush 12
3  ip netns exec rout2 ip r flush 21
4  ip netns exec rout2 ip r flush 22
5
6  ip netns exec rout1 ip rule add from 192.168.100.0/24 table 11
7  ip netns exec rout1 ip r add prohibit 192.168.200.0/24 \
8      from 192.168.100.0/24 table 11
9  ip netns exec rout1 ip r add default via 172.16.1.254 table 11
10
11 ip netns exec rout1 ip rule add from 192.168.200.0/24 table 12
12 ip netns exec rout1 ip r add prohibit 192.168.100.0/24 \
13     from 192.168.200.0/24 table 12
14 ip netns exec rout1 ip r add default via 172.16.1.254 table 12
15
16 ip netns exec rout2 ip rule add from 192.168.100.0/24 table 21
17 ip netns exec rout2 ip r add prohibit 192.168.200.0/24 \
18     from 192.168.100.0/24 table 21
19 ip netns exec rout2 ip r add default via 172.16.2.254 table 21
20
21 ip netns exec rout2 ip rule add from 192.168.200.0/24 table 22
22 ip netns exec rout2 ip r add prohibit 192.168.100.0/24 \
23     from 192.168.200.0/24 table 22
24 ip netns exec rout2 ip r add default via 172.16.2.254 table 22
```

Listing 1.12: Forbid traffic between different VLAN with Policy Routing

As we can see the result in figure 11, we cannot ping from `poste1` to `poste2` after applying the above Policy Routing. When using the target `prohibit`, we receive the message `Packet Filtered`. If we use the target `unreachable`, the returned message is `Destination Host Unreachable`.

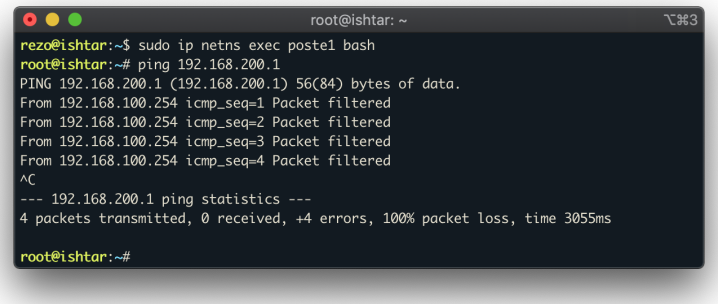


Fig. 11: Forbid packets from VLAN100 to VLAN200

9 Conclusion

In this project, we implemented a network with “trunking” VLAN. We studied protocol L2TPv3 to build a tunnel with different modes, then compared the results. We also compared protocol L2TPv3

with VXLAN and GRE. IPsec was also analysed in this project. Finally, we connect the VLANs to the Internet and use `iptables` as well as Policy Routing to forbid traffic between different VLANs. For redirect traffic from `poste1` to the Internet via `rout2`, we add one more DHCP server on `rout2`.

References

1. Ieee standard for local and metropolitan area networks–media access control (mac) security, https://1.ieee802.org/security/802-1ae/
2. Ferro, G.: Overlay networking vxlan means mpls in the data centre is dead (2013), https://etherealmind.com/overlay-networking-vxlan-means-mpls-in-the-data-centre-is-dead/?doing_wp_cron=1589102747.4892740249633789062500
3. RFC2661: Layer two tunneling protocol "l2tp", https://tools.ietf.org/html/rfc2661
4. RFC3931: Layer two tunneling protocol - version 3 (l2tpv3), https://tools.ietf.org/html/rfc3931
5. RFC7348: Virtual extensible local area network (vxlan): A framework for overlaying virtualized layer 2 networks over layer 3 networks, https://tools.ietf.org/html/rfc7348