



**Université
de Limoges**

UNIVERSITY OF LIMOGES

Faculty of Science and Technology

Master 1 CRYPTIS

**Sécurité de l'Information et Cryptologie
(CRYPTIS)**

Parcours Informatique

Projet Infrastructure Réseaux - Semestre II

Tunnel L2TPv3 sécurisé par IPsec

MAKHOUL Vladimir

SALAME Joe

Encadrant

M. Conchon Emmanuel

M. Bonnefoi Pierre-Francois

15 mai 2023

Table des matières

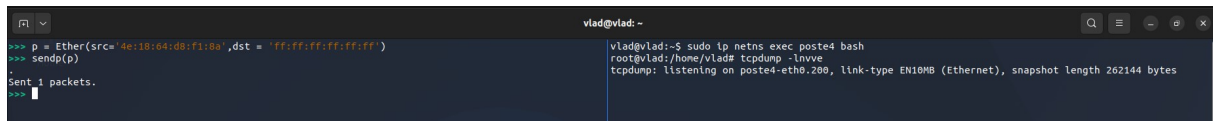
1	VLAN	2
2	L2TPv3 et VXLANs	4
2.1	L2TPv3	4
2.2	VXLAN	5
2.3	L2TPv3 vs VXLAN :	5
2.4	Sécurité	6
2.5	Comparaison avec MPLS :	6
3	Tunnel L2TPv3 en mode encapsulation IP	7
3.1	Configuration du tunnel	7
3.2	ARP entre 2 postes	7
3.3	Configuration du DHCP	8
3.4	Connexion TCP avec Socat	9
4	Comparaison entre GRE et L2tpv3 : IP, UDP	10
4.1	L2tpv3 encapsulation IP	10
4.2	L2tpv3 encapsulation UDP	11
4.3	GRE mode GRE-TAP	12
5	IPsec	13
5.1	Configuration de IPsec	13
5.2	Comparaison avec Iperf	15
6	Accès Internet	16
6.1	Configuration du switch et de la machine réelle	16
6.2	Configuration de l'accès à Internet sur Routeur1 pour les postes	17
6.3	Le trafic de Poste1 passe bien par Routeur1	18
6.4	Redirigition des postes 1 et 2 vers routeur 2	19
7	Isolation des Vlan	21
7.1	Iptables	21
7.2	Policy Routing	22

Chapitre 1

VLAN

Tout d'abord nous avons créé les netnamespaces : les routeurs, les switches et les postes, puis nous avons configuré les routeurs et les vlans associés à chaque interface, après cela nous avons configuré les postes, enfin nous avons créé les routes et ajouté les default gateways.

Comme vous pouvez le voir sur cette image, nous avons créé un paquet en utilisant scapy avec le mac source du poste3 et l'adresse de broadcast comme destination et nous avons envoyé le paquet, et nous avons exécuté tcpdump sur le poste4. Nous pouvons constater que le paquet n'a pas été reçu car ils sont sur des vlans différents.



```
vlad@vlad: ~  
>>> p = Ether(src='4e:1b:64:d8:f1:8a',dst = 'ff:ff:ff:ff:ff:ff')  
>>> sendp(p)  
>  
Sent 1 packets.  
>>>  
vlad@vlad:~$ sudo ip netns exec poste4 bash  
root@vlad:/home/vlad# tcpdump -lnvve  
tcpdump: listening on poste4-eth0.200, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

Dans cet exemple poste 3 appartenant à vlan100 envoie un ping 192.168.100.254 qui est l'adresse de l'interface rout1-eth0.100 spécifiée pour vlan 100

Comme indiqué dans la capture d'écran, le tcpdump affiche l'en-tête Ethernet avec un ethertype 802.1Q vlan 100. de plus l'arp envoyé avant l'icmp n'est capturé que par l'interface rout1-eth0.100 qui appartient au même vlan les autres appartenant au vlan 200 n'ont rien capturé qui indique que les lan sont correctement séparés

```

Vlad@vlad:~$ sudo ip netns exec rout1 tcpdump -i rout1-eth0 -lnvvee 'ip or icmp or arp'
tcpdump: listening on rout1-eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
15:26:35.810866 4e:18:64:d8:f1:8a > ff:ff:ff:ff:ff:ff, ethertype 802.1Q (0x8100), length 46: vlan 1
00, p 0, ethertype ARP (0x0806), Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.100.254 te
ll 192.168.100.2, length 28
15:26:35.810882 d6:65:b5:17:14:db > 4e:18:64:d8:f1:8a, ethertype 802.1Q (0x8100), length 46: vlan 1
00, p 0, ethertype ARP (0x0806), Ethernet (len 6), IPv4 (len 4), Reply 192.168.100.254 is-at d6:65:
b5:17:14:db, length 28
15:26:35.811099 4e:18:64:d8:f1:8a > d6:65:b5:17:14:db, ethertype 802.1Q (0x8100), length 102: vlan
100, p 0, ethertype IPv4 (0x0800), (tos 0x0, ttl 64, id 49455, offset 0, flags [DF], proto ICMP (1)
, length 84)
192.168.100.2 > 192.168.100.254: ICMP echo request, id 459, seq 1, length 64
15:26:35.811114 d6:65:b5:17:14:db > 4e:18:64:d8:f1:8a, ethertype 802.1Q (0x8100), length 102: vlan
100, p 0, ethertype IPv4 (0x0800), (tos 0x0, ttl 64, id 29821, offset 0, flags [none], proto ICMP (
2), length 64)
192.168.100.254 > 192.168.100.2: ICMP echo reply, id 459, seq 1, length 64
15:26:40.895356 d6:65:b5:17:14:db > 4e:18:64:d8:f1:8a, ethertype 802.1Q (0x8100), length 46: vlan 1
00, p 0, ethertype ARP (0x0806), Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.100.2 tell
192.168.100.254, length 28
15:26:40.895542 4e:18:64:d8:f1:8a > d6:65:b5:17:14:db, ethertype 802.1Q (0x8100), length 46: vlan 1
00, p 0, ethertype ARP (0x0806), Ethernet (len 6), IPv4 (len 4), Reply 192.168.100.2 is-at 4e:18:64:
d8:f1:8a, length 28

Vlad@vlad:~$ sudo ip netns exec poste4 tcpdump -lnvvee 'ip or icmp or arp'
tcpdump: listening on poste4-eth0.200, link-type EN10MB (Ethernet), snapshot length 262144 bytes

Vlad@vlad:~/lnfra_proj5$ sudo ip netns exec poste3 ping 192.168.100.254 -c 1
PING 192.168.100.254 (192.168.100.254) 56(84) bytes of data:
64 bytes from 192.168.100.254: icmp_seq=1 ttl=64 time=0.560 ms
--- 192.168.100.254 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/ndev = 0.560/0.560/0.560/0.000 ms
Vlad@vlad:~/lnfra_proj5$

Vlad@vlad:~$ sudo ip netns exec rout1 tcpdump -i rout1-eth0.200 -lnvvee 'ip or icmp or arp'
tcpdump: listening on rout1-eth0.200, link-type EN10MB (Ethernet), snapshot length 262144 bytes

```

Dreoulment des echanges :

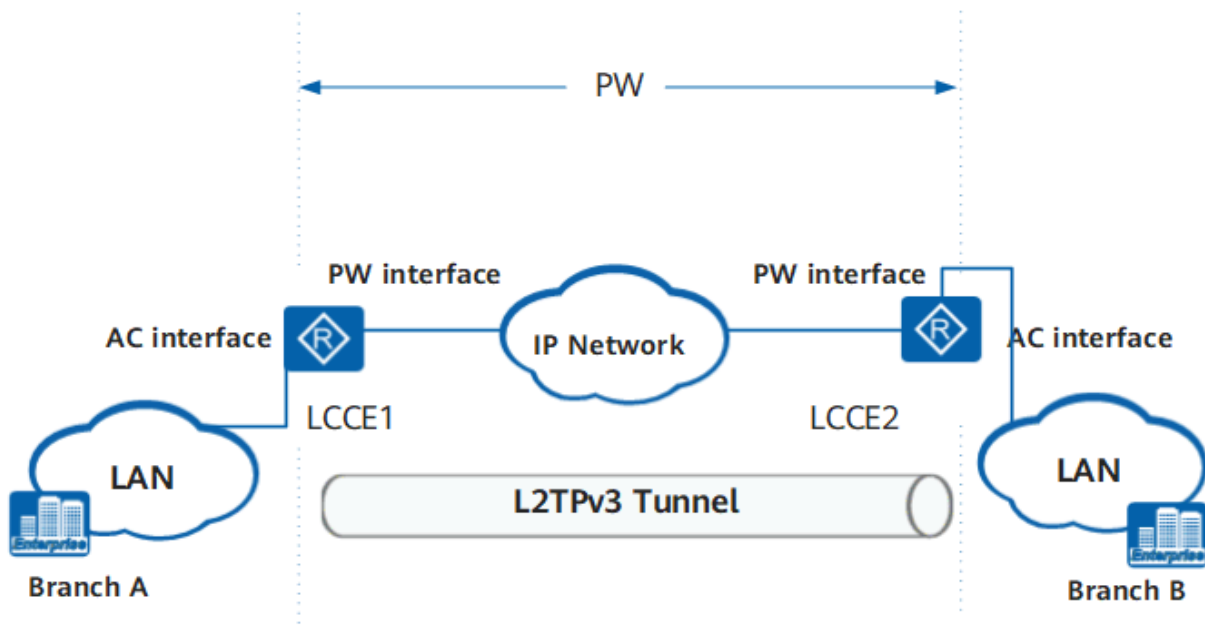
- **poste3** mac : 4e :18 :64 :d8 :f1 :8a -- > **rout1-eth0.100** mac : d6 :65 :b5 :17 :14 :db

Chapitre 2

L2TPv3 et VXLANs

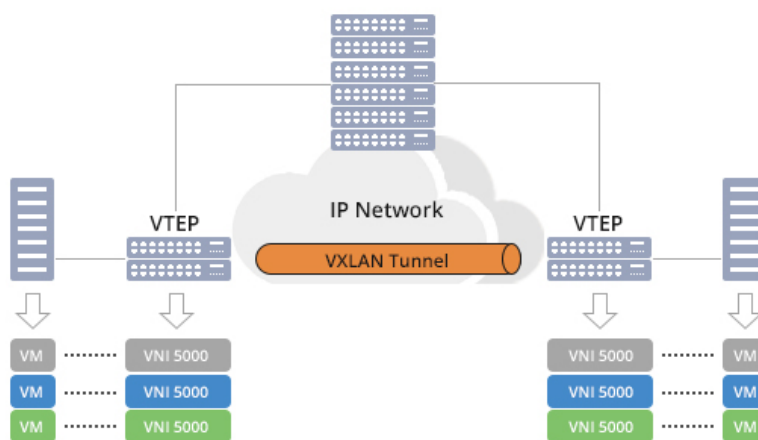
2.1 L2TPv3

L2TPv3 est un protocole de tunnellation de couche 2 utilisé pour transporter le trafic de couche 2 sur un réseau IP. Il encapsule Ethernet ou d'autres trames de L2 dans des paquets IP, qui sont ensuite transmis sur le réseau. Le protocole L2TPv3 est couramment utilisé pour connecter différents wide area networks (WAN), tels que des branch offices ou des data centers. Des tunnels L2TPv3 peuvent être établis entre des routeurs, des switches ou d'autres périphériques réseau, permettant au trafic de L2 d'être transmis de manière transparente entre eux. L2TPv3 peut être configuré avec différentes options, telles que la taille MTU, le cryptage et l'authentification, pour optimiser ses performances et sa sécurité.



2.2 VXLAN

VXLAN est un protocole de tunneling qui tunnelise le trafic de la couche 2 de l'Ethernet sur un réseau de la couche 3 de l'IP. Il s'agit d'une extension du réseau local virtuel (vlan) et d'un protocole de couche d'application basé sur udp qui fonctionne sur le port 4789. VXLAN surmonte les problèmes des réseaux traditionnels de la couche 2, tels que le spanning tree, le nombre limité de vlans et les grandes tables d'adresses mac. Il utilise unlogical network identifier de 24 bits, ce qui permet d'augmenter le nombre de vlans et d'isoler davantage le réseau logique pour les grands réseaux tels que les nuages. La technologie VXLAN permet de créer jusqu'à 16 millions de VXLAN dans un domaine administratif, ce qui permet la migration de machines virtuelles entre des serveurs qui existent dans des domaines de couche 2 distincts.



2.3 L2TPv3 vs VXLAN :

L2TPv3 (Layer 2 Tunneling Protocol version 3) et VXLAN (Virtual Extensible LAN) sont deux protocoles qui permettent de créer des tunnels pour transporter des paquets Ethernet sur un réseau IP. Toutefois, ils ont des différences notables :

- **Portée du réseau :** L2TPv3 est conçu pour les déploiements en WAN (Wide Area Network) tandis que VXLAN est conçu pour les déploiements en DC (Data Center).
- **Encapsulation :** L2TPv3 encapsule les paquets Ethernet dans des paquets UDP (User Datagram Protocol) alors que VXLAN utilise une encapsulation UDP-GRE (Generic Routing Encapsulation) pour encapsuler les paquets Ethernet.
- **Overhead :** L'encapsulation L2TPv3 est plus légère que celle de VXLAN, ce qui entraîne un surcoût moindre en bande passante.
- **Scalabilité :** VXLAN est plus évolutif que L2TPv3 en termes de nombre de VLANs (Virtual Local Area Networks) et de la taille du réseau.
- **Fonctionnalités :** VXLAN est plus riche en fonctionnalités que L2TPv3, avec notamment la possibilité d'isoler des trafics avec des réseaux virtuels (VRF) et la prise en charge de la qualité de service (QoS).

En résumé, L2TPv3 est plus adapté pour les déploiements WAN tandis que VXLAN est plus adapté pour les déploiements en Data Center, grâce à sa meilleure évolutivité et ses fonctionnalités plus avancées.

2.4 Sécurité

L2TPv3 peut être utilisé avec plusieurs protocoles de chiffrement, notamment IPSec, SSL/TLS, SSH, PPTP et L2TP. Cela signifie que les données L2TPv3 peuvent être chiffrées de bout en bout, offrant une sécurité accrue pour le trafic de couche 2.

D'autre part, VXLAN ne prend pas en charge le chiffrement natif et nécessite généralement une autre solution de chiffrement, telle que IPSec, pour sécuriser le trafic. Cela signifie que VXLAN est plus flexible en termes de choix des solutions de chiffrement, mais peut nécessiter plus de configuration pour être sécurisé.

Le choix entre L2TPv3 et VXLAN dépend des besoins spécifiques de l'environnement de réseau. Si la sécurité du trafic est une préoccupation majeure, L2TPv3 peut être la meilleure option. Si la flexibilité et la facilité de configuration sont plus importantes, VXLAN peut être préférable.

2.5 Comparaison avec MPLS :

MPLS :

Utilise un étiquetage de couche 2 (ou 3) pour acheminer les paquets dans le réseau. Fournit une qualité de service (QoS) grâce à la classification des paquets en fonction de leur type de trafic et à la mise en place de politiques de traitement des paquets. Les routeurs MPLS doivent maintenir des tables de routage spéciales contenant des informations sur les étiquettes MPLS. Est souvent utilisé dans les réseaux WAN (Wide Area Network) pour connecter des succursales distantes.

L2TPv3 :

Encapsule les trames Ethernet de couche 2 dans des paquets IP de couche 3. Permet de connecter des réseaux locaux distants via un tunnel sécurisé. Supporte les protocoles de routage IP tels que OSPF et BGP pour l'acheminement des paquets. Ne fournit pas nativement de QoS.

VXLAN : Encapsule les trames Ethernet de couche 2 dans des paquets UDP de couche 4. Permet de créer des réseaux logiques de grande échelle dans des centres de données virtualisés. Prend en charge les fonctionnalités de routage et de commutation du protocole Ethernet. Permet de fournir une QoS via la classification des paquets et la définition de politiques de traitement des paquets. Peut être utilisé en combinaison avec des technologies de virtualisation telles que VMWare et OpenStack.

En termes de performance, MPLS est généralement considéré comme le plus rapide et le plus efficace en termes de routage. L2TPv3 et VXLAN sont tous deux des solutions plus flexibles qui peuvent être utilisées dans des scénarios spécifiques tels que la connexion de réseaux distants ou la virtualisation de centres de données.

Chapitre 3

Tunnel L2TPv3 en mode encapsulation IP

3.1 Configuration du tunnel

La même configuration pour le routeur 2 mais nous avons commuté pour la première ligne l'adresse du point de terminaison local à remote et vice-versa même pour le *tunnel_id* et le *peer_{tunnel_id}*

```
1 #tunnel rout1
2 ip netns exec rout1 ip l2tp add tunnel remote 172.16.2.253 local 172.16.1.253
   encap ip tunnel_id 3000 peer_tunnel_id 4000
3 ip netns exec rout1 ip l2tp add session tunnel_id 3000 session_id 1000
   peer_session_id 2000
4 ip netns exec rout1 ip link set l2tpeth0 up
5 ip netns exec rout1 brctl addbr tunnel
6 ip netns exec rout1 brctl addif tunnel l2tpeth0
7 ip netns exec rout1 ip link set dev l2tpeth0 mtu 1500
8 ip netns exec rout1 brctl addif tunnel rout1-eth1
9 ip netns exec rout1 ip link set tunnel up
10 ip netns exec rout1 ip link add link tunnel name tunnel.100 type vlan id 100
11 ip netns exec rout1 ip link set tunnel.100 up
12 ip netns exec rout1 ip link add link tunnel name tunnel.200 type vlan id 200
13 ip netns exec rout1 ip link set tunnel.200 up
14 ip netns exec rout1 ip addr add 192.168.100.254/24 dev tunnel.100
15 ip netns exec rout1 ip addr add 192.168.200.254/24 dev tunnel.200
```

3.2 ARP entre 2 postes

Après avoir implémenté le tunnel L2tpv3, nous avons envoyé un ping de poste4 à poste2 qui est de l'autre côté d'internet et nous pouvons voir la requête arp envoyée avant la requête icmp capturée par le tcpdump sur le poste2. De plus avant l'echo-reply, poste2 a envoyé une autre requête arp pour trouver le mac de poste4 et il a obtenu la réponse de poste4 comme le montre l'image.

3.3. CONFIGURATION DU DHCP3. TUNNEL L2TPV3 EN MODE ENCAPSULATION IP

```
vlad@vlad:~$ sudo ip netns exec poste1 ping 192.168.200.1 -c 1
PING 192.168.200.1 (192.168.200.1) 64(04) bytes of data:
64 bytes from 192.168.200.1: icmp_seq=1 ttl=64 time=0.857 ms

--- 192.168.200.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.857/0.857/0.857/0.000 ms
vlad@vlad:~$
```

```
vlad@vlad:~$ sudo ip netns exec poste2 tcpdump -lnvve -i poste2-eth0 'arp or ip'
tcpdump: listening on poste2-eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
00:25:22.983267 fa:3f:f3:31:ff:6d > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Ethernet (I
en 6), IPv4 (Len 4), Request who-has 192.168.200.1 tell 192.168.200.2, length 28
00:25:22.983289 32:82:5f:04:97:e2 > fa:3f:f3:31:ff:6d, ethertype ARP (0x0806), length 42: Ethernet (I
en 6), IPv4 (Len 4), Reply 192.168.200.1 is-at 32:82:5f:04:97:e2, length 28
00:25:22.983330 fa:3f:f3:31:ff:6d > 32:82:5f:04:97:e2, ethertype IPv4 (0x0800), length 98: (tos 0x0,
ttl 64, id 23946, offset 0, flags [DF], proto ICMP (1), length 84)
192.168.200.2 > 192.168.200.1: ICMP echo request, id 36163, seq 1, length 64
00:25:22.983543 32:82:5f:04:97:e2 > fa:3f:f3:31:ff:6d, ethertype IPv4 (0x0800), length 98: (tos 0x0,
ttl 64, id 6746, offset 0, flags [none], proto ICMP (1), length 84)
192.168.200.1 > 192.168.200.2: ICMP echo reply, id 36163, seq 1, length 64
00:25:28.133952 32:82:5f:04:97:e2 > fa:3f:f3:31:ff:6d, ethertype ARP (0x0806), length 42: Ethernet (I
en 6), IPv4 (Len 4), Request who-has 192.168.200.2 tell 192.168.200.1, length 28
00:25:28.134232 fa:3f:f3:31:ff:6d > 32:82:5f:04:97:e2, ethertype ARP (0x0806), length 42: Ethernet (I
en 6), IPv4 (Len 4), Reply 192.168.200.2 is-at fa:3f:f3:31:ff:6d, length 28
```

3.3 Configuration du DHCP

```
1 #DHCP
2 ip netns exec rout1 dnsmasq -d -z -i tunnel.100 --except-interface=lo -F
   192.168.100.1,192.168.100.10,255.255.255.0 -l /tmp/dnsmasq.leases &
3 ip netns exec rout1 dnsmasq -d -z -i tunnel.200 --except-interface=lo -F
   192.168.200.1,192.168.200.10,255.255.255.0 -l /tmp/dnsmasq.leases &
4
5 #DHCP request
6 ip netns exec poste1 dhclient &
7 ip netns exec poste2 dhclient &
8 ip netns exec poste4 dhclient &
9 ip netns exec poste3 dhclient &
```

Après avoir exécuté la commande dhclient -v dans le poste1, il a commencé à chercher un serveur dhcp en envoyant en diffusion "DHCPDISCOVER", quand il a trouvé le serveur dhcp il a envoyé une requête à ce serveur qui a répondu par 'DHCPOFFER' offrant à poste1 la plage d'ips disponibles. Ensuite poste1 a choisi le ip '192.168.100.63'. Nous avons ajouté la commande dhclient dans le script afin qu'il puisse obtenir l'adresse IP automatiquement.

```
vlad@vlad:~$ sudo ip netns exec rout2 ip addr add 192.168.100.253/24 dev tunnel.100
+ ip netns exec rout2 ip addr add 192.168.200.253/24 dev tunnel.200
+ ip netns exec rout1 dnsmasq -d -z -i tunnel.100 -f 192.168.100.1,192.168.100.10,255.255.255.0 -l /
tmp/dnsmasq.leases --dhcp-opts=3,192.168.100.253
dnsmasq: started, version 2.80 cachesize 150
dnsmasq: compile time options: IPV6 GNU-getopt DBus no-UBUS libn IDN2 DHCP DHCPv6 no-Lua TFTP contra
ck ipset auth cryptohash DNSSEC loop-detect inotify dumpfile
dnsmasq-dhcp: DHCP, IP range 192.168.100.1 -- 192.168.100.100, lease time 1h
dnsmasq-dhcp: DHCP, sockets bound exclusively to interface tunnel.100
dnsmasq: reading /etc/resolv.conf
dnsmasq: using nameserver 127.0.0.53#53
dnsmasq: read /etc/hosts - 7 addresses
dnsmasq-dhcp: not giving name vlad to the DHCP lease of 192.168.100.4 because the name exists in /etc
/hosts with address 127.0.1.1
dnsmasq-dhcp: DHCPDISCOVER(tunnel.100) 192.168.100.2 02:87:ad:4b:f4:27
dnsmasq-dhcp: DHCPOFFER(tunnel.100) 192.168.100.63 02:87:ad:4b:f4:27
dnsmasq-dhcp: DHCPDISCOVER(tunnel.100) 192.168.100.3 02:87:ad:4b:f4:27
dnsmasq-dhcp: DHCPOFFER(tunnel.100) 192.168.100.63 02:87:ad:4b:f4:27
dnsmasq-dhcp: DHCPREQUEST(tunnel.100) 192.168.100.63 02:87:ad:4b:f4:27
dnsmasq-dhcp: DHCPACK(tunnel.100) 192.168.100.63 02:87:ad:4b:f4:27 vlad
dnsmasq-dhcp: not giving name vlad to the DHCP lease of 192.168.100.63 because the name exists in /et
c/hosts with address 127.0.1.1

loop txqueuelen 1000 (Local Loopback)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

poste1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet6 fe80::87:adff:fe4b:f427 prefixlen 64 scopeid 0x20<link>
ether 02:87:ad:4b:f4:27 txqueuelen 1000 (Ethernet)
RX packets 129 bytes 12434 (12.4 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 15 bytes 1262 (1.2 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

poste1-eth0.100: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet6 fe80::87:adff:fe4b:f427 prefixlen 64 scopeid 0x20<link>
ether 02:87:ad:4b:f4:27 txqueuelen 1000 (Ethernet)
RX packets 22 bytes 1540 (1.5 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

vlad@vlad:~$
```

```
vlad@vlad:~$ sudo ip netns exec poste1 dhclient -v
Internet Systems Consortium DHCP Client 4.4.1
Copyright 2004-2018 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/poste1-eth0/02:87:ad:4b:f4:27
Sending on LPF/poste1-eth0/02:87:ad:4b:f4:27
Listening on LPF/poste1-eth0.100/02:87:ad:4b:f4:27
Sending on LPF/poste1-eth0.100/02:87:ad:4b:f4:27
Sending on Socket/fallback
DHCPDISCOVER on poste1-eth0 to 255.255.255.255 port 67 interval 3 (xid=0xcdb56972)
DHCPREQUEST for 192.168.100.2 on poste1-eth0.100 to 255.255.255.255 port 67 (xid=0x555f7661)
DHCPDISCOVER on poste1-eth0 to 255.255.255.255 port 67 interval 0 (xid=0xcdb56972)
DHCPREQUEST for 192.168.100.2 on poste1-eth0.100 to 255.255.255.255 port 67 (xid=0x555f7661)
DHCPDISCOVER on poste1-eth0 to 255.255.255.255 port 67 interval 11 (xid=0xcdb56972)
DHCPDISCOVER on poste1-eth0.100 to 255.255.255.255 port 67 interval 3 (xid=0xb1ad721e)
DHCPDISCOVER on poste1-eth0.100 to 255.255.255.255 port 67 interval 8 (xid=0xb1ad721e)
DHCPOFFER for 192.168.100.63 from 192.168.100.254
DHCPREQUEST for 192.168.100.63 on poste1-eth0.100 to 255.255.255.255 port 67 (xid=0x1e72ad81)
DHCPACK of 192.168.100.63 from 192.168.100.254 (xid=0xb1ad721e)
bound to 192.168.100.63 -- renewal in 1771 seconds.
vlad@vlad:~$
```

```
Poste1
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

Avant DHCP
Après DHCP
```

3.4 Connexion TCP avec Socat

Avant de créer la connexion socat entre poste1 et route1, nous avons exécuté tcpdump dans l'interface tunnel.100 des deux côtés du tunnel (routeur1 et routeur2) pour voir si le trafic passe par le tunnel. Ensuite, lors de l'établissement de la connexion TCP entre le routeur1 et le poste1, nous pouvons voir des deux côtés le TCP Handshake (SYN-SYN/ACK-ACK) échangée entre eux dans le tcpdump sur l'interface tunnel.100 du routeur1 et l'interface tunnel.100 du routeur2 indiquant qu'il a réussi à traverser le tunnel.

```

Route1 tunnel.100
Route2 tunnel.100

```

Nous avons également vérifié si le trafic passait par le tunnel en envoyant un message

```

Route1 tunnel.100
Route2 tunnel.100

```

Comparaison entre GRE et L2tpv3 : IP, UDP

Ici, nous pouvons voir le paquet IP capturé dans scapy, il utilise le protocole l2tp. On peut également remarquer que la longueur du paquet est de 70. Les sources IP et la destination de rout1 et rout2 se trouvent dans l'en-tête IP. Les sources IP encapsulées et la destination peuvent être trouvées dans le payload (la trame ARP encapsulé), C0A86401 : 192.168.100.1(poste1) et C0A86402 : 192.168.100.2(poste3)

10

4.2 L2tpv3 encapsulation UDP

Pour le mode d'encapsulation udp, nous avons un en-tête ip d'une longueur de 138 qui est presque 2 fois plus grand que le paquet ip du mode d'encapsulation ip. La source et la destination ip dans l'en-tête ip sont les mêmes mais le proto est maintenant udp car c'est le mode d'encapsulation udp. l'en-tête UDP contient un port source de 1234, un port de destination de 4321 et une longueur de 118. nous pouvons également voir les adresses des adresses postel et post3 dans le chargement (l'ARP encapsulé).

[illegible]


4.3 GRE mode GRETA

```
1 ip netns exec rout1 ip link add eoip1 type gretap remote 172.16.2.253 local
    172.16.1.253 nopmtudisc
2 ip netns exec rout1 ip link set dev eoip1 up
```

Pour l'encapsulation GRE mode GRE-TAP nous avons dans l'entête ip le protocole gre, longueur 70 et les adresses source et destination des interfaces de router1 et router2. Dans l'en-tête GRE, le proto est TEB (Transparent Ethernet Bridging Protocol) qui indique que le paquet encapsulé dans le tunnel GRE est une trame Ethernet et est ponté de manière transparente à travers le tunnel. nous pouvons également voir que la trame encapsulée est lisible, nous avons la trame Ethernet avec le 802.3Q qui est pour le vlan dans cette image c'est le vlan 100. Après l'en-tête 802.3Q, nous voyons le contenu ARP indiquant que 192.168.100.1 demandant quel mac a l'adresse 192.168.100.2

```
vlad@vlad: ~  
>>> pkt = sniff(count=1, filter='lo')  
>>> pkt[1].show()  
###[ Ethernet II ]##  
    dst= ee:88:43:68:b5:2b  
    src= ee:88:50:39:18:fc  
    type= IPv4  
###[ IP ]##  
    version= 4  
    ihl= 5  
    tos= 0x0  
    len= 76  
    id= 6901  
    flags= 0  
    frag= 0  
    ttl= 63  
    protocol= GRE  
    checksum= 0x37a  
    src= 172.16.1.233  
    dst= 172.16.1.233  
    window= 0  
###[ GRE ]##  
    checksum_present= 0  
    routing_present= 0  
    key_present= 0  
    sequen_present= 0  
    strict_route_source= 0  
    recursion_control= 0  
    flags= 0  
    version= 0  
    protocol= TEER  
###[ Ethernet II ]##  
    dst= ff:ff:ff:ff:ff:ff  
    src= de:88:97:52:18:3a  
    type= n_802_1Q  
###[ 802.1Q ]##  
    prio= 0  
    len= 0  
    vlan= 100  
    type= ARP  
###[ ARP ]##  
    htype= 0x1  
    ptype= 0x1  
    hwlen= 6  
    plen= 4  
    op= who-has  
    source= de:88:97:52:18:3a  
    source= 192.168.100.1  
    dest= de:88:97:52:18:3a  
    dest= 192.168.100.2  
>>>  
vlad@vlad: ~  
>>> p = Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst="192.168.100.1")  
>>> send(p)  
Sent 1 packets.  
>>>
```

GRETAP (Generic Routing Encapsulation over Tunneling Protocol) est un protocole de tunnellation utilisé pour encapsuler des trames Ethernet dans des paquets IP. L'unité de transmission maximale (MTU) de GRETAP est de 1462 octets. D'autre part, le MTU de L2TPv3 peut varier en fonction de la configuration, mais le MTU par défaut est de 1500 octets. Ici, nous pouvons voir que le MTU est 1458. Par conséquent, le MTU de GRETAP est inférieur au MTU par défaut de L2TPv3. Cela signifie que si nous utilisons GRETAP, nous devrons peut-être ajuster le MTU de nos interfaces et périphériques réseau pour éviter les problèmes de fragmentation et de performances.



```
vlad@vlad:~$ sudo ip netns exec routi tfconfig | grep tunnel | grep mtu
tunnel: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1458
tunnel:100: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1458
tunnel:200: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1458
vlad@vlad:~$
```

```
vlad@vlad:~$ sudo ip netns exec routi tfconfig | grep tunnel | grep mtu
tunnel: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1462
tunnel:100: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1462
tunnel:200: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1462
vlad@vlad:~$
```

Chapitre 5

IPsec

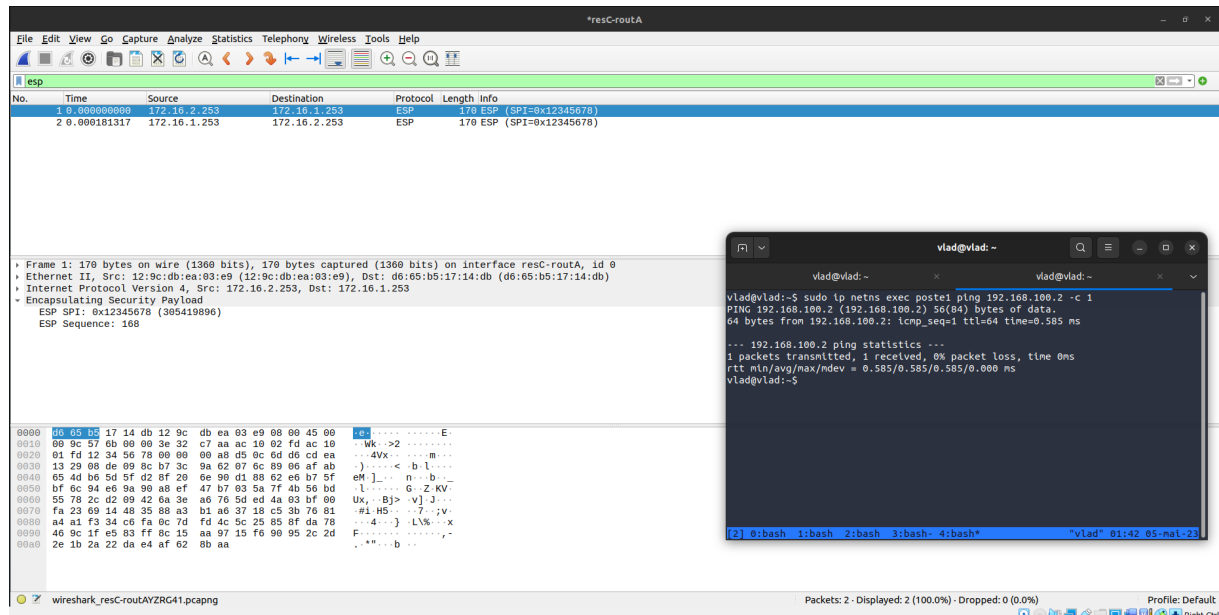
IPsec (Internet Protocol Security) est un protocole de sécurité utilisé pour sécuriser les communications sur un réseau IP. Il offre des mécanismes de chiffrement, d'authentification et d'intégrité des données pour garantir la confidentialité et l'intégrité des informations échangées entre les différents périphériques sur un réseau. L2TPv3 seul ne fournit pas de mécanismes de sécurité, il est donc souvent combiné avec IPsec pour ajouter une couche de sécurité aux communications. Lorsqu'IPsec est utilisé avec L2TPv3, le trafic de données est d'abord encapsulé dans des paquets L2TPv3, puis ces paquets sont à leur tour encapsulés dans des paquets IPsec. Cela permet de protéger le trafic de données à la fois au niveau de la couche 2 (avec L2TPv3) et au niveau de la couche IP (avec IPsec).

5.1 Configuration de IPsec

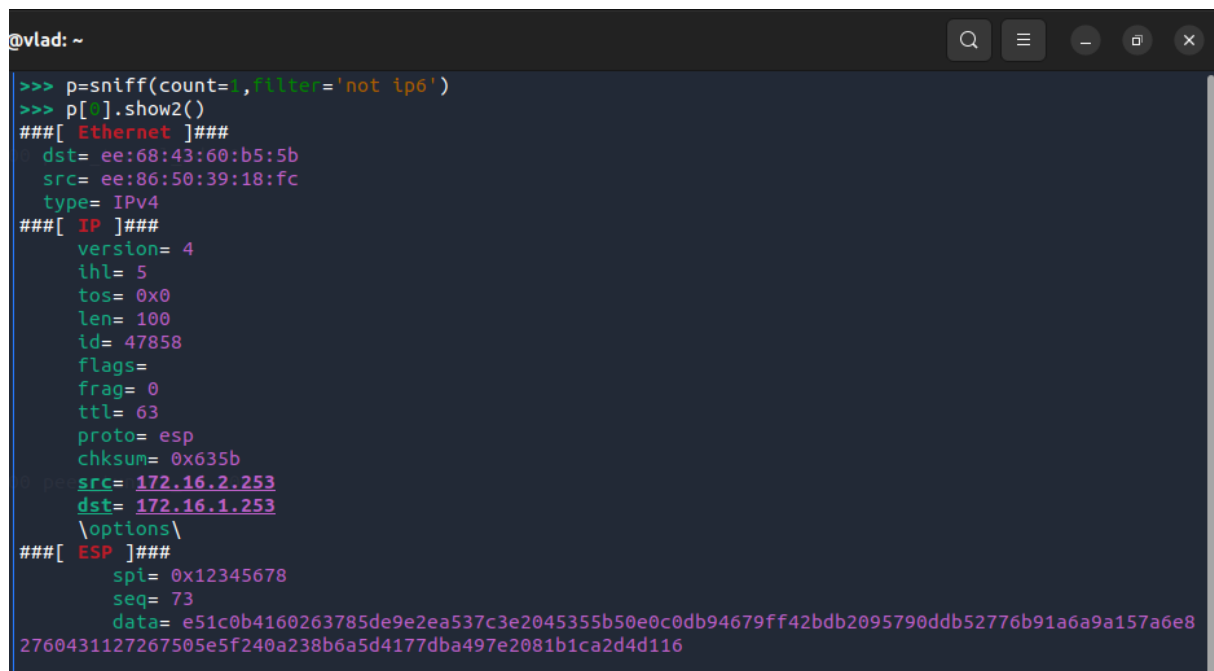
Pour ajouter ipsec nous avons ajouté à notre script les configurations suivantes :

```
1 #rout1
2 ip netns exec rout1 ip xfrm state flush
3 ip netns exec rout1 ip xfrm policy flush
4 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 0
   x323730ed6f1b9ff0cb084af15b197e862b7c18424a7cdfb74cd385ae23bc4f17 enc "
   rf3686(ctr(aes))" 0x27b90b8aec1ee32a8150a664e8faac761e2d305b
5 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 0
   x44d65c50b7581fd3c8169cf1fa0ebb24e0d55755b1dc43a98b539bb144f2067f enc "
   rf3686(ctr(aes))" 0x9df7983cb7c7eb2af01d88d36e462b5f01d10bc1
6 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
7 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
```

cette image de capture wireshark qui montre le protocole ESP (Encapsulating Security Payload) utilisé dans ipsec pour assurer la confidentialité, l'intégrité et l'authentification des paquets transférés sur les réseaux IP, ce qui indique que la configuration fonctionne.

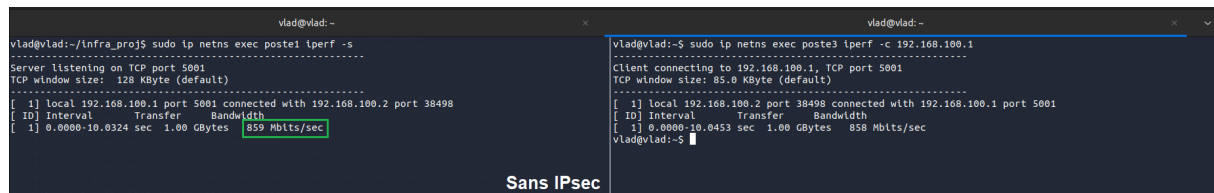


Dans l'image ci-dessous, nous avons utilisé Scapy pour sniffer le paquet ipsec afin de l'afficher sous une forme plus lisible



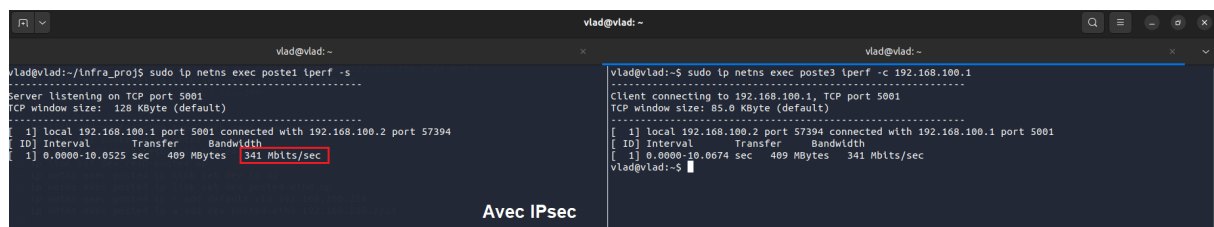
5.2 Comparaison avec Iperf

Avant d'ajouter la configuration ipsec nous avons mesuré la vitesse de transfert sans ipsec en utilisant iperf pour la comparer plus tard avec la vitesse avec ipsec. La première image sans ipsec a donné une vitesse de 859 Mbits/sec tandis que dans la deuxième image (avec ipsec) elle a donné 341 Mbits/sec ce qui est 2,5 fois plus lent que la vitesse sans ipsec cela est dû à une cause de cryptage, contrôle d'intégrité et un processus d'authentification qui prend beaucoup de temps mais qui offre un haut niveau de sécurité.



```
vlad@vlad:~$ sudo ip netns exec post1 iperf -s
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
-----
[ 1] local 192.168.100.1 port 5001 connected with 192.168.100.2 port 38498
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-10.0324 sec  1.00 GBytes  859 Mbits/sec
vlad@vlad:~$
```

Sans IPsec



```
vlad@vlad:~$ sudo ip netns exec post1 iperf -s
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
-----
[ 1] local 192.168.100.1 port 5001 connected with 192.168.100.2 port 57394
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-10.0525 sec  409 MBytes  341 Mbits/sec
vlad@vlad:~$
```

Avec IPsec

Chapitre 6

Accès Internet

6.1 Configuration du switch et de la machine réelle

Nous avons commencé par configurer un pont réseau qui permettra la connexion des netnamespaces de se connecter à Internet en reliant les interfaces réseau des netnamespaces au pont, le trafic peut être acheminé entre le réseau externe via le pont.

```
1 #Cr ation d'un pont r seau
2 sudo ip link add br0 type bridge
3
4 #Attribution du pont r seau l'interface root-eth0
5 sudo ip link set root-eth0 master br0
6
7 #Attribution d'une adresse IP l'interface br0
8 sudo ip address add 192.168.100.6/24 dev br0
9
10 #Activation de l'interface br0
11 sudo ip link set br0 up
12
13 #Configuration des r gles du firewall pour le forwarding du trafic
14 iptables -A FORWARD -o enp0s3 -i root-eth0 -j ACCEPT
15 iptables -A FORWARD -i enp0s3 -o root-eth0 -j ACCEPT
16
17 #Configuration de la translation d'adresses r seau (NAT) pour le trafic sortant
18 iptables -t nat -A POSTROUTING -s 10.87.0.254/24 -o enp0s3 -j MASQUERADE
```

Après nous avons relié l'interface net-root du netnamespace au switch OVS, et en activant l'interface root-eth0 du côté de l'hôte, nous avons créé un pont entre les netnamespace et l'hôte principal. Les étapes supplémentaires d'activation de l'interface loopback (lo) et de configuration d'une adresse IP à l'interface root-eth0 contribuent à la connectivité réseau. Le switch OVS, dans ce cas nommé "internet", agit comme un commutateur virtuel qui peut faire le lien entre les différentes interfaces réseau. L'interface net-root est ajoutée comme un port au switch OVS, ce qui lui permet de se connecter à d'autres interfaces réseau via le switch.

```
1 #Cr ation d'une paire de interfaces virtuelles veth
2 ip link add root-eth0 type veth peer name net-root
3 #Configuration de l'interface root-eth0
4 ip link set root-eth0
5 #Ajout de l'interface net-root au switch OVS
6 ovs-vsctl add-port internet net-root
7 #Activation des interfaces net-root, root-eth0 et loopback
8 ip link set dev net-root up
9 ip link set dev root-eth0 up
10 ip link set dev lo up
11 #Attribution d'une adresse IP à l'interface root-eth0
12 ip a add dev root-eth0 10.87.0.254/24
13 #Activation du forwarding (routage)
14 sysctl net.ipv4.conf.all.forwarding=1
```

6.2 Configuration de l'accès à Internet sur Routeur1 pour les postes

Maintenant, pour permettre aux postes de se connecter à Internet, nous avons utilisé les règles de nat suivantes que nous avons créées sur le routeur 1

```
1 ip netns exec rout1 iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -j
   MASQUERADE -o rout1-eth0
2 ip netns exec rout1 iptables -t nat -A POSTROUTING -s 192.168.200.0/24 -j
   MASQUERADE -o rout1-eth0
3 ip netns exec routA iptables -t nat -A POSTROUTING -s 172.16.1.0/24 -j
   MASQUERADE -o routA-eth1
4 ip netns exec routB iptables -t nat -A POSTROUTING -s 172.16.2.0/24 -j
   MASQUERADE -o routB-eth1
5 ip netns exec rout2 iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -j
   MASQUERADE -o rout2-eth0
6 ip netns exec rout2 iptables -t nat -A POSTROUTING -s 192.168.200.0/24 -j
   MASQUERADE -o rout2-eth0
```

Nous pouvons voir dans cette image un ping fonctionnel de chaque poste vers Internet

```
vlad@vlad: ~  
rtt min/avg/max/mdev = 13.772/14.881/16.589/1.225 ms  
vlad@vlad:~/infra_proj$ clear  
vlad@vlad:~/infra_proj$ sudo ip netns exec poste1 ping 8.8.8.8 -c 3  
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=116 time=14.8 ms  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=13.5 ms  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=14.2 ms  
  
--- 8.8.8.8 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2007ms  
rtt min/avg/max/mdev = 13.498/14.185/14.811/0.537 ms  
vlad@vlad:~/infra_proj$ sudo ip netns exec poste2 ping 8.8.8.8 -c 3  
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=116 time=14.2 ms  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=14.4 ms  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=14.6 ms  
  
--- 8.8.8.8 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2016ms  
rtt min/avg/max/mdev = 14.193/14.393/14.606/0.168 ms  
vlad@vlad:~/infra_proj$ sudo ip netns exec poste3 ping 8.8.8.8 -c 3  
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=116 time=13.8 ms  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=14.6 ms  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=13.9 ms  
  
--- 8.8.8.8 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2019ms  
rtt min/avg/max/mdev = 13.827/14.121/14.633/0.363 ms  
vlad@vlad:~/infra_proj$ sudo ip netns exec poste4 ping 8.8.8.8 -c 3  
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=116 time=14.4 ms  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=14.8 ms  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=13.9 ms  
  
--- 8.8.8.8 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2002ms  
rtt min/avg/max/mdev = 13.925/14.378/14.775/0.349 ms  
vlad@vlad:~/infra_proj$
```

6.3 Le trafic de Poste1 passe bien par Routeur1

Comme nous avons configuré le DHCP uniquement sur le routeur 1, une passerelle par défaut de .254 sera attribuée pour tous les postes, c'est pourquoi le trafic du poste 1 passera par le routeur 1.

```
vlad@vlad:~$ sudo ip netns exec poste1 route  
Kernel IP routing table  
Destination Gateway Genmask Flags Metric Ref Use Iface  
default 192.168.100.254 0.0.0.0 UG 0 0 0 poste1-eth0.100  
192.168.100.0 0.0.0.0 255.255.255.0 U 0 0 0 poste1-eth0.100  
vlad@vlad:~$ sudo ip netns exec poste2 route  
Kernel IP routing table  
Destination Gateway Genmask Flags Metric Ref Use Iface  
default 192.168.200.254 0.0.0.0 UG 0 0 0 poste2-eth0.200  
192.168.200.0 0.0.0.0 255.255.255.0 U 0 0 0 poste2-eth0.200  
vlad@vlad:~$ sudo ip netns exec poste3 route  
Kernel IP routing table  
Destination Gateway Genmask Flags Metric Ref Use Iface  
default 192.168.100.254 0.0.0.0 UG 0 0 0 poste3-eth0.100  
192.168.100.0 0.0.0.0 255.255.255.0 U 0 0 0 poste3-eth0.100  
vlad@vlad:~$ sudo ip netns exec poste4 route  
Kernel IP routing table  
Destination Gateway Genmask Flags Metric Ref Use Iface  
default 192.168.200.254 0.0.0.0 UG 0 0 0 poste4-eth0.200  
192.168.200.0 0.0.0.0 255.255.255.0 U 0 0 0 poste4-eth0.200  
vlad@vlad:~$  
vlad@vlad:~$ sudo ip netns exec poste1 traceroute 8.8.8.8  
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets  
1 192.168.100.254 (192.168.100.254) 0.111 ms 0.027 ms 0.022 ms  
2 172.16.1.254 (172.16.1.254) 0.033 ms 0.028 ms 0.026 ms  
3 10.87.0.254 (10.87.0.254) 0.038 ms 0.032 ms 0.031 ms  
4 192.168.100.1 (192.168.100.1) 0.385 ms 0.331 ms 0.294 ms^C  
vlad@vlad:~$
```

6.4 Redirigation des postes 1 et 2 ver routeur 2

Pour optimiser l'accès à internet nous allons configurer un DHCP sur le routeur 2 à l'aide de ces commandes :

```
1 ip netns exec rout2 dnsmasq -d -z -i tunnel.100 --except-interface=lo -F
   192.168.100.11,192.168.100.20,255.255.255.0 -l /tmp/dnsmasq.leases &
2
3 ip netns exec rout2 dnsmasq -d -z -i tunnel.200 --except-interface=lo -F
   192.168.200.11,192.168.200.20,255.255.255.0 -l /tmp/dnsmasq.leases &
```

Ces commandes permettent à dnsmasq d'agir en tant que serveur DHCP, fournissant des adresses IP aux clients qui se connectent aux interfaces "tunnel.100" et "tunnel.200" du réseau namespace "rout2".

Nous pouvons également allouer une plage de seulement 2 ips dans chaque commande "dnsmasq" pour empêcher les postes de l'autre côté du tunnel d'obtenir les ips de l'autre serveur DHCP car elles seront déjà prises par les appareils directement connectés à celui-ci. Nous pou-

vons maintenant voir que poste1 et poste 2 obtiennent leur adresse IP du routeur 2 au lieu du routeur 1

```
vlad@vlad: ~/infra_proj$ sudo ip netns exec poste1 route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
default 192.168.100.253 0.0.0.0 UG 0 0 0 poste1-eth0.100
192.168.100.0 0.0.0.0 255.255.255.0 U 0 0 0 poste1-eth0.100
vlad@vlad: ~/infra_proj$ sudo ip netns exec poste2 route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
default 192.168.200.253 0.0.0.0 UG 0 0 0 poste2-eth0.200
192.168.200.0 0.0.0.0 255.255.255.0 U 0 0 0 poste2-eth0.200
vlad@vlad: ~/infra_proj$ sudo ip netns exec poste3 route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
default 192.168.100.254 0.0.0.0 UG 0 0 0 poste3-eth0.100
192.168.100.0 0.0.0.0 255.255.255.0 U 0 0 0 poste3-eth0.100
vlad@vlad: ~/infra_proj$ sudo ip netns exec poste4 route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
default 192.168.200.254 0.0.0.0 UG 0 0 0 poste4-eth0.200
192.168.200.0 0.0.0.0 255.255.255.0 U 0 0 0 poste4-eth0.200
vlad@vlad: ~/infra_proj$
```

Nous pouvons voir plus clairement dans ce traceroute que le trafic de poste1 et poste2 est passé par le routeur 2 à destination internet

```
vlad@vlad: ~/infra_proj$ sudo ip netns exec poste1 route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
default          192.168.100.253 0.0.0.0          UG    0      0      0 poste1-eth0.100
192.168.100.0    0.0.0.0          255.255.255.0    U      0      0      0 poste1-eth0.100
vlad@vlad: ~/infra_proj$ sudo ip netns exec poste2 route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
default          192.168.200.253 0.0.0.0          UG    0      0      0 poste2-eth0.200
192.168.200.0    0.0.0.0          255.255.255.0    U      0      0      0 poste2-eth0.200
vlad@vlad: ~/infra_proj$ sudo ip netns exec poste3 route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
default          192.168.100.254 0.0.0.0          UG    0      0      0 poste3-eth0.100
192.168.100.0    0.0.0.0          255.255.255.0    U      0      0      0 poste3-eth0.100
vlad@vlad: ~/infra_proj$ sudo ip netns exec poste4 route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
default          192.168.200.254 0.0.0.0          UG    0      0      0 poste4-eth0.200
192.168.200.0    0.0.0.0          255.255.255.0    U      0      0      0 poste4-eth0.200
vlad@vlad: ~/infra_proj$
```

Chapitre 7

Isolation des Vlan

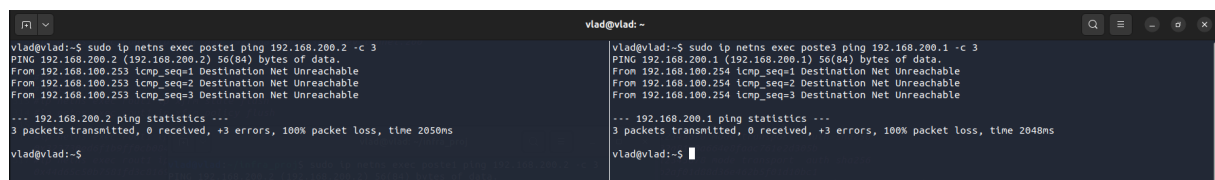
7.1 Iptables

Pour interdire la communication entre vlan100 et vlan200 en utilisant les règles iptables. Nous avons écrit sur les deux routeurs rout1 et rout2 les règles suivantes :

```
1 ip netns exec rout1 iptables -t filter -A FORWARD -s 192.168.100.0/24 -d
  192.168.200.0/24 -j REJECT --reject-with icmp-net-unreachable
2 ip netns exec rout1 iptables -t filter -A FORWARD -s 192.168.200.0/24 -d
  192.168.100.0/24 -j REJECT --reject-with icmp-net-unreachable
3 ip netns exec rout2 iptables -t filter -A FORWARD -s 192.168.100.0/24 -d
  192.168.200.0/24 -j REJECT --reject-with icmp-net-unreachable
4 ip netns exec rout2 iptables -t filter -A FORWARD -s 192.168.200.0/24 -d
  192.168.100.0/24 -j REJECT --reject-with icmp-net-unreachable
```

Si un paquet provenant d'un appareil appartenant au réseau '192.168.100.0/24'(appareils vlan100) veut communiquer avec un autre appareil qui appartient au réseau '192.168.200.0/24'(appareils vlan200) ou vice versa ce paquet sera rejeté et un paquet icmp indiquant que le réseau est inaccessible sera renvoyé à l'appareil qui a envoyé le paquet.

Par exemple, nous pouvons voir sur l'image que nous envoyons un ping de poste1 à poste4 et de de poste3 à poste2. Nous recevons le message "Destination Net Unreachable"



```
vlad@vlad:~$ sudo ip netns exec poste1 ping 192.168.200.1 -c 3
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data:
From 192.168.100.253 icmp_seq=1 Destination Net Unreachable
From 192.168.100.253 icmp_seq=2 Destination Net Unreachable
From 192.168.100.253 icmp_seq=3 Destination Net Unreachable
--- 192.168.200.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2050ms
vlad@vlad:~$

vlad@vlad:~$ sudo ip netns exec poste3 ping 192.168.200.1 -c 3
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data:
From 192.168.100.254 icmp_seq=1 Destination Net Unreachable
From 192.168.100.254 icmp_seq=2 Destination Net Unreachable
From 192.168.100.254 icmp_seq=3 Destination Net Unreachable
--- 192.168.200.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2048ms
vlad@vlad:~$
```

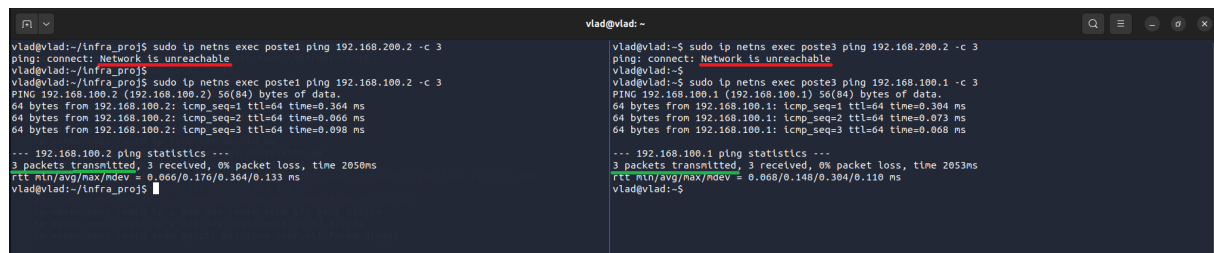
7.2 Policy Routing

Maintenant pour interdire la communication en utilisant `ip Policy Routing`, nous avons écrit les règles suivantes sur `route1` et `route2` :

```
1 ip netns exec route1 ip rule add from 192.168.100.0/24 to 192.168.200.0/24
  blackhole
2 ip netns exec route1 ip rule add from 192.168.200.0/24 to 192.168.100.0/24
  blackhole
3 ip netns exec route2 ip rule add from 192.168.100.0/24 to 192.168.200.0/24
  blackhole
4 ip netns exec route2 ip rule add from 192.168.200.0/24 to 192.168.100.0/24
  blackhole
```

Ces règles vérifient si la source et la destination appartiennent à des vlans différents (192.168.100.0/24 [Vlan100] et 192.168.200.0/24 [Vlan200]) elles seront rejetées. le 'blackhole' spécifie qu'aucun paquet icmp ne sera renvoyé à l'appareil auquel est envoyé ce paquet pour communiquer avec l'appareil avec un vlan différent

Par exemple, nous pouvons voir sur l'image que nous envoyons un ping de `poste1` à `poste4` et de `poste3` à `poste2`. Nous recevons "Network is Unreachable"



```
vlad@vlad:~/infra_proj$ sudo ip netns exec poste1 ping 192.168.200.2 -c 3
ping: connect: Network is unreachable
vlad@vlad:~/infra_proj$
vlad@vlad:~/infra_proj$ sudo ip netns exec poste1 ping 192.168.100.2 -c 3
PING 192.168.100.2 (192.168.100.2) 56(84) bytes of data:
64 bytes from 192.168.100.2: icmp_seq=1 ttl=64 time=0.364 ms
64 bytes from 192.168.100.2: icmp_seq=2 ttl=64 time=0.066 ms
64 bytes from 192.168.100.2: icmp_seq=3 ttl=64 time=0.098 ms
--- 192.168.100.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2050ms
rtt min/avg/max/mdev = 0.066/0.176/0.364/0.133 ms
vlad@vlad:~/infra_proj$

vlad@vlad:~/infra_proj$ sudo ip netns exec poste3 ping 192.168.200.2 -c 3
ping: connect: Network is unreachable
vlad@vlad:~/infra_proj$
vlad@vlad:~/infra_proj$ sudo ip netns exec poste3 ping 192.168.100.1 -c 3
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data:
64 bytes from 192.168.100.1: icmp_seq=1 ttl=64 time=0.304 ms
64 bytes from 192.168.100.1: icmp_seq=2 ttl=64 time=0.073 ms
64 bytes from 192.168.100.1: icmp_seq=3 ttl=64 time=0.068 ms
--- 192.168.100.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2053ms
rtt min/avg/max/mdev = 0.068/0.148/0.304/0.110 ms
vlad@vlad:~/infra_proj$
```