



Université  
de Limoges

## **Master 1 : CRYPTIS Informatique**

---

# **Rapport projet Infrastructure réseau : Tunnel L2TPv3 sécurisé par IPsec pour la mise en œuvre de « trunking » de VLANs et comparaison avec les VXLANs**

---

*Réalisé par :*

Jean DE BONFILS

Sarra JLASSI

Version du  
14 mai 2021

## Table des matières

<b>1</b>	<b>Présentation du projet</b>	<b>2</b>
<b>2</b>	<b>Architecture du réseau et VLANs</b>	<b>3</b>
2.1	Architecture de départ . . . . .	3
2.2	Ajouts des VLANs . . . . .	3
<b>3</b>	<b>Présentation du protocole L2TPv3 et de la technologie des VXLANs</b>	<b>6</b>
3.1	Présentation et comparaison des deux solutions . . . . .	6
3.2	Mise en œuvre dans Linux des VXLANs dans Open vSwitch . . . .	6
3.3	Chiffrement du trafic L2TPv3 ou VXLAN . . . . .	7
3.4	Comparaison avec MPLS . . . . .	7
<b>4</b>	<b>Établissement du tunnel L2TPv3</b>	<b>9</b>
4.1	Mise en place du tunnel en mode encapsulation IP et UDP . . . . .	9
4.1.1	En mode IP . . . . .	9
4.1.2	En mode UDP . . . . .	10
4.2	Vérification de la mise en place . . . . .	10
4.3	Comparaison du mode d'encapsulation IP et UDP . . . . .	14
4.3.1	Encapsulation IP . . . . .	14
4.3.2	Encapsulation UDP . . . . .	15
4.4	Mise en place d'un tunnel GRE en mode GRE-TAP . . . . .	16
4.5	Comparaison des tunnels L2TPv3 et GRE . . . . .	18
<b>5</b>	<b>Mise en place de chiffrement avec IPsec sur le tunnel</b>	<b>20</b>
<b>6</b>	<b>Accès à internet "intelligent"</b>	<b>24</b>
6.1	Accès à internet via le routeur 1 . . . . .	24
6.2	Redirection avec dnsmasq des postes 1 et 2 vers le routeur 2 pour optimiser l'accès . . . . .	28
<b>7</b>	<b>Blocage des VLANs</b>	<b>30</b>
7.1	A l'aide du firewall . . . . .	30
7.2	A l'aide de la "Policy Routing" . . . . .	30
<b>8</b>	<b>Organisation du projet</b>	<b>34</b>

## 1 Présentation du projet

Le but de ce projet consiste à réaliser un tunnel de niveau 2 avec le protocole **L2TPV3**. Ce tunnel est utilisé pour faire du **trunking** entre deux VLANs. Ensuite, on s'en servira pour faire de la mutualisation de service comme le DHCP. Plusieurs modes de fonctionnement pourront être utilisés, notamment, l'utilisateur pourra choisir entre une encapsulation IP ou UDP ainsi qu'un tunnel L2TPv3 ou GRE. Du chiffrement grâce à IPsec sera mis en place sur le tunnel. Enfin, on limitera l'utilisation du tunnel lorsque les hôtes voudront se connecter à Internet en leur permettant de le faire directement de leur **côté d'Internet** à l'aide d'une configuration **intelligente**. Des règles de Firewalls et Policy Routing seront mises en place pour éviter la communication entre deux VLANs différents.

## 2 Architecture du réseau et VLANs

### 2.1 Architecture de départ

Pour la mise en place du réseau, on s'est inspiré des différents TP's déjà réalisés. Donc on a commencé par créer le fichier script shell **build\_architecture** qui permet de construire l'architecture de départ. Tout d'abord on a créé les différents netns grâce à la commande **ip netns add**. Ensuite, on a attribué à chaque netns une adresse IP statique grâce à la commande **ip a add dev**. Par la suite, on a configuré les routes et les routes par défaut à l'aide des commandes **ip route add** et **ip route add default**. Enfin, on a activé le routage à l'aide de la commande **sudo sysctl net.ipv4.conf.all.forwarding=1**.

Après la mise en place des différents netns, on a généré le graphe correspondant à l'architecture réseau de départ à l'aide de la commande suivante :

**python3 network\_graph.py > graph.dot dot -Tpng graph.dot -o graph.png**  
comme le montre la figure suivante :

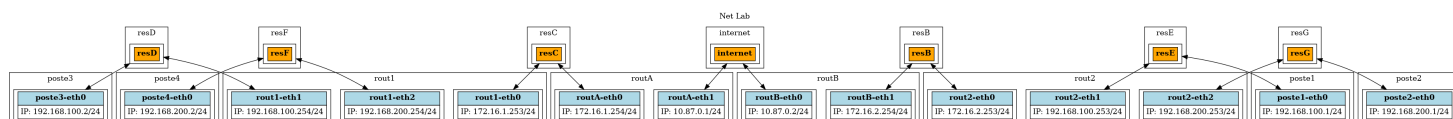


FIGURE 1 – Architecture de départ

Ensuite, pour la mise en œuvre de l'architecture du réseau correspondant au schéma demandé où les réseaux sont liés au niveau 2, on a créé le fichier script shell **build\_all** permettant de lancer tous les scripts qui modifient l'architecture de départ et permettant ainsi d'obtenir l'architecture de réseau final (pour plus d'informations sur le script "build\_all" voir ch.8).

### 2.2 Ajouts des VLANs

Les VLANs ont été créés afin que tous les netns puissent communiquer entre eux s'ils sont connectés au même bridge (internet) et appartiennent au même VLAN. Pour cela, on a créé le fichier script shell **VLANs** qui permet d'ajouter des VLANs de port sur les interfaces des netns pour permettre le «trunking» au trafic étiqueté au travers de tunnel de niveau 2. Donc, tout d'abord, on a commencé par supprimer les adresses IP de la configuration initiale des différents netns grâce à la commande **ip a del dev**. Ensuite, on a créé de "sous" interfaces (poste1-eth0.100, poste2-eth0.200, poste3-eth0.100, poste4-eth0.200) au niveau

des interfaces (poste1-eth0, poste2-eth0, poste3-eth0, poste4-eth0). Puis on les a ajouté dans les VLANs correspondants avec la commande : **ip link add link poste[i]-eth0 name poste[i]-eth0.100/200 type vlan id 100/200** avec i le numéro de poste.

Afin de tester le bon fonctionnement de l'encapsulation VLAN, on a capturé du trafic au niveau du routeur 1. Ce trafic correspond à un ping (ICMP) entre le poste 1 et le poste 3.

```
jean@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns$ sudo ip netns exec rout1 tcpdump -lnvve -i rout1-eth1
tcpdump: listening on rout1-eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
10:39:48.634813 6a:ea:24:b9:64:87 > ea:26:ed:97:8e:a2, ethertype 802.1Q (0x8100), length 102: vlan 100, p 0, e
thertype IPv4, (tos 0x0, ttl 64, id 6691, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.100.1 > 192.168.100.2: ICMP echo request, id 17909, seq 1, length 64
10:39:48.634813 6a:ea:24:b9:64:87 > ea:26:ed:97:8e:a2, ethertype 802.1Q (0x8100), length 102: vlan 100, p 0, e
thertype IPv4, (tos 0x0, ttl 64, id 53070, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.100.2 > 192.168.100.1: ICMP echo reply, id 17909, seq 1, length 64
10:39:49.634856 6a:ea:24:b9:64:87 > ea:26:ed:97:8e:a2, ethertype 802.1Q (0x8100), length 102: vlan 100, p 0, e
thertype IPv4, (tos 0x0, ttl 64, id 6891, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.100.1 > 192.168.100.2: ICMP echo request, id 17909, seq 2, length 64
10:39:49.634920 6a:ea:24:b9:64:87 > ea:26:ed:97:8e:a2, ethertype 802.1Q (0x8100), length 102: vlan 100, p 0, e
thertype IPv4, (tos 0x0, ttl 64, id 53146, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.100.2 > 192.168.100.1: ICMP echo reply, id 17909, seq 2, length 64
```

FIGURE 2 – Capture de paquets échangés dans une VLAN

A partir de la capture de trafic VLAN ci-dessus, on peut en déduire qu'il s'agit bien des trames échangées entre le poste 1 et le poste 3. Les adresses Mac correspondent bien aux adresses des deux postes comme le justifient les captures ci-dessous. On observe également le tag VLAN 100 ce qui est conforme à nos attentes. Le datagramme IP contient du trafic ICMP provenant de l'adresse IP 192.168.100.1 et à destination de l'adresse IP 192.168.100.2. On peut vérifier que ces adresses MAC et IP correspondent bien aux adresses des postes 1 et 3 grâce aux captures ci-dessous :

```
poste1-eth0.100: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.100.1 netmask 255.255.255.0 broadcast 0.0.0.0
    inet6 fe80::68ea:24ff:feb9:6487 prefixlen 64 scopeid 0x20<link>
    ether 6a:ea:24:b9:64:87 txqueuelen 1000 (Ethernet)
    RX packets 50 bytes 3416 (3.4 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 30 bytes 2580 (2.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns#
```

FIGURE 3 – Adresses IP et MAC sur l'interface poste1-eth0.100 du poste 1

```
poste3-eth0.100: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.100.2 netmask 255.255.255.0 broadcast 0.0.0.0
    inet6 fe80::e826:edff:fe97:8ea2 prefixlen 64 scopeid 0x20<link>
    ether ea:26:ed:97:8e:a2 txqueuelen 1000 (Ethernet)
    RX packets 50 bytes 3416 (3.4 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 30 bytes 2580 (2.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns#
```

**FIGURE 4** – Adresses IP et MAC sur l'interface poste3-eth0.100 du poste 3

## 3 Présentation du protocole L2TPv3 et de la technologie des VXLANs

### 3.1 Présentation et comparaison des deux solutions

Le protocole L2TPv3 (**Layer 2 Tunneling Protocol Version 3**) est un protocole d'encapsulation point à point d'un protocole quelconque de niveau 2 dans des datagrammes IP. Il fournit un mécanisme dynamique pour faire du tunneling sur la couche 2 à travers un réseau de données en mode paquet (comme par exemple IP). Ce protocole est une évolution importante de L2TPv2 qui n'autorise que l'encapsulation du protocole PPP (**Point to Point Protocol**).

La technologie des VXLANs, l'acronyme de (**Virtual eXtensible LAN**) est une technologie de virtualisation réseau qui utilise une technique d'encapsulation proche des Vlan et permet d'encapsuler le trafic de la couche 2 dans des datagrammes UDP.

Ces deux solutions permettent toutes les deux à transiter des trames de niveau 2 sur un réseau IP. Elles sont souvent utilisées avec le protocole IPsec qui permet le chiffrement des données. Cependant, le protocole L2TPv3 utilise l'encapsulation IP et la technologie des VXLANs qui utilise l'encapsulation UDP. Aussi, pour faire passer le trafic provenant des vlans dans un tunnel utilisant le protocole L2TPv3, on a besoin d'étiquetter chaque vlan au niveau des interfaces du tunnel. Les VLANS offrent une capacité de 4096 machines au maximum par VLAN contrairement à VXLAN qui permet d'avoir plus de machines dans une même VLAN.

### 3.2 Mise en œuvre dans Linux des VXLANs dans Open vSwitch

Pour la mise en place des VXLANs dans Open vSwitch sous Linux, il faut exécuter les commandes suivantes au niveau de chaque extrémité du tunnel :

- On commence tout d'abord par créer le switch br0 à l'aide de la commande **ovs-vsctl add-br br0**.
- Ensuite, on ajoute l'interface vxlan et l'adresse ip correspondante, aussi on peut ajouter plusieurs options comme modifier le port par défaut de destination avec **options** : à l'aide de la commande **ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=vxlan options :remote\_ip=AAA.BBB.CCC.DDD**

### 3.3 Chiffrement du trafic L2TPv3 ou VXLAN

Le protocole L2TPv3 et la technologie VXLAN sont souvent utilisés avec un protocole de chiffrement qui est le protocole **IPsec**. Ce protocole permet de chiffrer les données du trafic et permet de créer ainsi des VPNs sécurisés. Il propose deux protocoles :

- Le protocole AH «**Authentication Header**» pour l'authentification. Il assure l'intégrité et l'authentification de l'origine pour l'ensemble des champs de l'entête du datagramme IP, à l'exception de ceux qui peuvent changer lors du transfert du datagramme, c-à-d les champs «TTL» et «checksum».
- Le protocole ESP «**Encapsulating Security Payload**» pour le chiffrement et l'authentification. Il permet de chiffrer le trafic et assure ainsi une sécurisation des paquets échangés. Il protège également contre les attaques par rejeu et garantit l'intégrité des données. On peut utiliser séparément l'un ou l'autre et, plus souvent, les deux ensembles.

Il propose également deux modes de fonctionnement :

- Le mode Transport : Il permet d'établir une liaison sécurisée directement entre deux matériels et d'encapsuler le chargement du datagramme IP : les @IP source et destination restent celles de ces matériels.
- Le mode Tunnel : Il permet d'établir une liaison sécurisée entre deux routeurs, d'encapsuler la totalité du datagramme IP passant par ces routeurs ce qui permet d'offrir un «securehop», c-à-d le passage sécurisé entre deux routeurs (les datagrammes IP ne contiennent que les @IP source et destination des routeurs ), mais pas celles des machines empruntant ce tunnel.

### 3.4 Comparaison avec MPLS

Le protocole MPLS (**MultiProtocol Label Switching**) est aussi une solution qui permet le transport de données basé sur des labels qui peuvent être insérés sur la couche 2 ou sur la couche 3 donc contrairement au protocole L2TPv3 et à la technologie VxLAN qui sont limités aux tunnels de niveau 2, le protocole MPLS permet de faire aussi bien des tunnels de niveau 2 que de niveau 3. Ainsi, il peut être utilisé pour transporter tout type de trafic tout en garantissant leur sécurité. Contrairement à L2TPv3 et VXLANs, MPLS est complètement séparé du reste du trafic et ne nécessite pas de chiffrement supplémentaire comme pour L2TPv3. De plus, contrairement à ces deux technologies, MPLS offre la



possibilité de réaliser de la QoS. Puisque le trafic est séparé, MPLS offre en théorie un meilleur débit ainsi qu'un jitter plus faible. De plus, le chiffrement mis en place sur des technologies telles que L2TPv3 réduit encore la bande passante pour ces technologies. Avec MPLS, il est notamment possible de rediriger le trafic et offrir un flux optimisé. Bien que MPLS propose de nombreux avantages, cette technologie reste beaucoup plus coûteuse à mettre en place que L2TPv3 ou VXLANs.

## 4 Établissement du tunnel L2TPv3

### 4.1 Mise en place du tunnel en mode encapsulation IP et UDP

#### 4.1.1 En mode IP

Pour établir le tunnel L2TPV3 en mode encapsulation IP, on a défini le fichier shell **tunnel\_L2TPv3\_IP** dans le dossier **scriptsTunnel** où on a créé le tunnel L2TPV3 entre le routeur 1 (rout1) et le routeur 2 (rout2) à l'aide de la commande **ip l2tp add tunnel remote 172.16.2.253 local 172.16.1.253 encap ip tunnel\_id 3000 peer\_tunnel\_id 4000/3000**. On a établi également la session du tunnel grâce à la commande **ip l2tp add session tunnel\_id 3000/4000 session\_id 1000 peer\_session\_id 2000**. Par la suite, on a activé les interfaces du tunnel des deux cotés à l'aide de **ip link set l2tpeth0 up**. Pour que le tunnel puisse acheminer des trames Ethernet, on a récupéré les trames à partir des interfaces rout1-eth1 et rout2-eth1 connectées aux VLANs. Pour cela, les interfaces rout1-eth1 et rout2-eth1 ont été accrochées au bridge "tunnel" créé avec la commande **brctl addbr tunnel**. Ce bridge permet de relier l'interface du routeur connecté aux réseaux. Par exemple, pour le routeur 1, l'interface rout1-eth1 est reliée à l'interface du tunnel ( l2tpeth0). Pour chaque routeur, on a ajouté le bridge ( tunnel) comme le montre la figure suivante :

```
#Pour R1
ip netns exec rout1 brctl addbr tunnel
ip netns exec rout1 brctl addif tunnel l2tpeth0
ip netns exec rout1 brctl addif tunnel rout1-eth1
ip netns exec rout1 brctl addif tunnel rout1-eth2
#Pour R2
ip netns exec rout2 brctl addbr tunnel
ip netns exec rout2 brctl addif tunnel l2tpeth0
ip netns exec rout2 brctl addif tunnel rout2-eth1
ip netns exec rout2 brctl addif tunnel rout2-eth2
```

**FIGURE 5** – Ajout du bridge tunnel

Ensuite, on a ajouté également les étiquettes VLAN ( vlan 100 ) et ( vlan 200 ) des interfaces ( tunnel.100 ) et ( tunnel.200 ) à l'interface tunnel représentant le bridge comme le montre la figure suivante :

```
#Pour R1
ip netns exec rout1 ip link add link tunnel name tunnel.100 type vlan id 100
ip netns exec rout1 ip link set tunnel.100 up
ip netns exec rout1 ip link add link tunnel name tunnel.200 type vlan id 200
ip netns exec rout1 ip link set tunnel.200 up
#Pour R2
ip netns exec rout2 ip link add link tunnel name tunnel.100 type vlan id 100
ip netns exec rout2 ip link set tunnel.100 up
ip netns exec rout2 ip link add link tunnel name tunnel.200 type vlan id 200
ip netns exec rout2 ip link set tunnel.200 up
```

**FIGURE 6** – Configuration des étiquettes VLAN et activation des interfaces

Au final, afin de permettre à la pile TCP/IP d'accéder aux réseaux des vlans, on a attribué des adresses IP aux interfaces du tunnel comme le montre la figure suivante :

```
ip netns exec rout1 ip addr add 192.168.100.254/24 dev tunnel.100
ip netns exec rout1 ip addr add 192.168.200.254/24 dev tunnel.200
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
```

**FIGURE 7** – Configuration IP de l'interface

#### 4.1.2 En mode UDP

Pour la mise en place du tunnel L2TPv3 en mode encapsulation UDP, on a créé le fichier shell **tunnel\_L2TPv3\_UDP** dans le dossier **scriptsTunnel**. La mise en place du tunnel en UDP est assez simple puisque on a utilisé pratiquement les mêmes commandes que celles pour la mise en place du tunnel utilisant de l'encapsulation IP.

## 4.2 Vérification de la mise en place

Pour vérifier la mise en place du tunnel L2TPv3, on a testé en premier lieu le fonctionnement du protocole ARP avec L2TPv3 en mode IP.

Pour cela, on a intercepté le trafic entre le poste 1 et le poste 3 à travers l'interface tunnel du routeur 1 ( rout1) à l'aide de la commande **tcpdump** comme le montre la figure suivante :

```

11:16:50.876192 IP (tos 0x0, ttl 64, id 52250, offset 0, flags [none], proto unknown (115), length 74)
  172.16.1.253 > 172.16.2.253: ip-proto-115 54
11:16:51.876417 IP (tos 0x0, ttl 62, id 1942, offset 0, flags [none], proto unknown (115), length 74)
  172.16.2.253 > 172.16.1.253: ip-proto-115 54
11:16:51.876444 IP (tos 0x0, ttl 64, id 52348, offset 0, flags [none], proto unknown (115), length 74)
  172.16.1.253 > 172.16.2.253: ip-proto-115 54
11:16:52.876707 IP (tos 0x0, ttl 62, id 2088, offset 0, flags [none], proto unknown (115), length 74)
  172.16.2.253 > 172.16.1.253: ip-proto-115 54
11:16:52.876774 IP (tos 0x0, ttl 64, id 52553, offset 0, flags [none], proto unknown (115), length 74)
  172.16.1.253 > 172.16.2.253: ip-proto-115 54

root@Jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns
Fichier Édition Affichage Rechercher Terminal Aide
^CSent 62 probes (1 broadcast(s))
Received 62 response(s)
root@Jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# arping -I poste1-eth0.100 192.168.100.2
ARPING 192.168.100.2 from 192.168.100.1 poste1-eth0.100
Unicast reply from 192.168.100.2 [4E:A2:61:55:44:45] 3.198ms
Unicast reply from 192.168.100.2 [4E:A2:61:55:44:45] 1.563ms
Unicast reply from 192.168.100.2 [4E:A2:61:55:44:45] 0.871ms
Unicast reply from 192.168.100.2 [4E:A2:61:55:44:45] 0.963ms
^CSent 4 probes (1 broadcast(s))
Received 4 response(s)

```

Nous avons également vérifié la mise en place du service DHCP sur le routeur 1. Pour cela, on a créé deux fichiers script shell, **DHCPServeur** et **AutoConfig\_Poste1\_DHCP** dans le dossier **scriptsDHCP** pour la configuration du DHCP. En effet, le fichier **DHCPServeur** permet la mise en place du serveur DHCP sur le routeur 1, en donnant une plage de 254 adresses allant de 1 à 254 pour les vlan100 et vlan200 à l'aide de la commande suivante :

**ip netns exec rout1 dnsmasq --interface=tunnel.100 --except-interface=lo --bind-interfaces --dhcp-range=192.168.100.1,192.168.100.254,255.255.255.0 --dhcp-option=option:router,192.168.100.253.** Quant au fichier **AutoConfig\_Poste1\_DHCP**, il permet de configurer dynamiquement l'adresse du poste 1 avec la commande **dhclient** après avoir supprimé son adresse statique à l'aide de la commande **ip netns exec poste1 ip a del** et qui avait été obtenu lors de la configuration initiale. Ci-dessous, on observe les différents messages du protocole DHCP.

```

DHCPDISCOVER on erspan0 to 255.255.255.255 port 67 interval 6 (xid=0x89f19f78)
send_packet: Network is down
dhclient.c:2438: Failed to send 300 byte long packet over erspan0 interface.
DHCPPREQUEST of 192.168.100.33 on poste1-eth0.100 to 255.255.255.255 port 67 (xid=0x2c706d2e)
DHCPOFFER of 192.168.100.33 from 192.168.100.254
DHCPACK of 192.168.100.33 from 192.168.100.254
cmp: EOF on /tmp/tmp.ax2hPVPJ2W which is empty
bound to 192.168.100.33 -- renewal in 1482 seconds.

```

On peut notamment observer le DHCP DISCOVER envoyé par le poste 1 et diffusé en broadcast. On observe également le DHCP OFFER qui est la proposition d'adresse IP du DHCP au poste 1. Ici, l'adresse 192.168.100.33 est proposée. En réponse, le poste 1 envoie le DHCP REQUEST permettant de formuler une

requête par rapport à l'adresse proposée. Enfin, un DHCP ACK est envoyé, il s'agit d'un accusé de réception.

Une fois le serveur DHCP lancé, on exécute le script **AutoConfig\_Poste1\_DHCP** comme le montre la figure suivante :

```

root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# ifconfig -a poste1-eth0.100
poste1-eth0.100: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.100.1 netmask 255.255.255.0 broadcast 0.0.0.0
    inet6 fe80::98d1:8fff:fe87:401d prefixlen 64 scopeid 0x20<link>
    ether 9a:d1:8f:87:40:1d txqueuelen 1000 (Ethernet)
    RX packets 39 bytes 2532 (2.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 14 bytes 1116 (1.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# sudo ./scriptsDHCP/AutoConfig_Poste1_DHCP &> out.log
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# ifconfig -a poste1-eth0.100
poste1-eth0.100: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.100.69 netmask 255.255.255.0 broadcast 192.168.100.255
    inet6 fe80::98d1:8fff:fe87:401d prefixlen 64 scopeid 0x20<link>
    ether 9a:d1:8f:87:40:1d txqueuelen 1000 (Ethernet)
    RX packets 46 bytes 3661 (3.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 20 bytes 3168 (3.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Avant, le poste 1 avait l'adresse IP **192.168.100.1**, maintenant une nouvelle adresse **192.168.100.69** lui a été attribué dynamiquement à partir du serveur DHCP s'exécutant sur le routeur 1.

Enfin, voici une preuve que le poste 1 se configure bien par le DHCP s'exécutant sur le routeur 1 comme demandé. En haut à gauche, le poste1 se configure par DHCP. En haut à droite on capture les paquets sortant du poste 1. En bas ,les configurations des interfaces du poste 1 et du routeur 1. On remarque que les adresses IP et MAC concordent bien et donc que le poste 1 se configure bien à partir d'un DHCP s'exécutant sur le routeur 1.

On a donc, **192.168.100.254** l'adresse IP du routeur 1 et **42 :68 :48 :1D :2C :05** son adresse Mac. **192.168.100.73** l'adresse IP proposée par le DHCP au poste 1. Et enfin l'adresse Mac du poste 1 **5A :CB :BE :0F :BB :4D**.

```

Jean@Jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns$ sudo ip netns exec rout1 dnsmasq --interface=tunnel.100 --except-interface=lo --bind-localhost --dhcp-range=192.168.100.1,192.168.100.254,255.255.255.0 --dhcp-option=option:router,192.168.100.253
Jean@Jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns/scriptsDHCP
root@Jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns/scriptsDHCP

Fichier Édition Affichage Rechercher Terminal Aide
DHCPDISCOVER on poste1-eth0 to 255.255.255.255 port 67 interval 3 (xid=0xd15f342)
DHCPDISCOVER on gretap0 to 255.255.255.255 port 67 interval 8 (xid=0x978e825f)
dhclient.c:2438: Failed to send 300 byte long packet over gretap0 interface.
DHCPDISCOVER on poste1-eth0 to 255.255.255.255 port 67 interval 5 (xid=0xd15f342)
DHCPDISCOVER on erspan0 to 255.255.255.255 port 67 interval 13 (xid=0x14a5237f)
dhclient.c:2438: Failed to send 300 byte long packet over erspan0 interface.
DHCPDISCOVER on poste1-eth0.100 to 255.255.255.255 port 67 interval 3 (xid=0x161d9a77)
DHCPDISCOVER on poste1-eth0 to 255.255.255.255 port 67 interval 12 (xid=0xd15f342)
DHCPDISCOVER on gretap0 to 255.255.255.255 port 67 interval 11 (xid=0x978e825f)
send_packet: Network is down
dhclient.c:2438: Failed to send 300 byte long packet over gretap0 interface.
DHCPREQUEST of 192.168.100.73 on poste1-eth0.100 to 255.255.255.255 port 67 (xid=0x779a1d16)
DHCPOFFER of 192.168.100.73 from 192.168.100.254
DHCPACK of 192.168.100.73 from 192.168.100.254
tmp: EOF on /tmp/tmp.4612zMM2nA which is empty
bound to 192.168.100.73 -- renewal in 1776 seconds.
Jean@Jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns/scriptsDHCP
root@Jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns/scriptsDHCP

Fichier Édition Affichage Rechercher Terminal Aide
tunnel.100: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1458
inet 192.168.100.254 netmask 255.255.255.0 broadcast 0.0.0.0
inet6 fe80::4068:48ff:fe1d:2c05 prefixlen 64 scopeid 0x20<link>
ether 42:68:48:1d:2c:05 txqueuelen 1000 (Ethernet)
RX packets 27 bytes 2708 (2.7 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 21 bytes 2043 (2.0 KB)

Fichier Édition Affichage Rechercher Terminal Aide
poste1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet6 fe80::58cb:beff:fe0f:bb4d prefixlen 64 scopeid 0x20<link>
ether 5a:cb:be:0f:bb:4d txqueuelen 1000 (Ethernet)
RX packets 190 bytes 18644 (18.6 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 44 bytes 8412 (8.4 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Après l'établissement du tunnel, on a effectué une connexion TCP avec la commande **socat** entre le routeur 1 ( mode serveur) et le poste 1 ( mode client) comme le montre la figure suivante :

```

Jean@Jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns$ sudo ip netns exec root@Jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns# sudo ip netns exec rout1
poste1 socat STDIO TCP:192.168.100.254:60567
root@Jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns# sudo ip netns exec rout1
socat TCP-LISTEN:60567,reuseaddr STDIO
bonjour
coucou
salut

```

```

Jean@Jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns
Fichier Édition Affichage Rechercher Terminal Aide
1:48:16.639586 ARP, Ethernet (len 6), IPv4 (len 4), Reply 172.16.1.253 is-at ea:a8:b7:8b:d8:8a, length 28
1:48:23.185875 IP (tos 0x0, ttl 62, id 17063, offset 0, flags [none], proto unknown (115), length 106)
172.16.2.253 > 172.16.1.253: ip-proto-115 86
1:48:23.186033 IP (tos 0x0, ttl 64, id 42207, offset 0, flags [none], proto unknown (115), length 98)
172.16.1.253 > 172.16.2.253: ip-proto-115 78
1:48:28.415017 IP (tos 0x0, ttl 64, id 43373, offset 0, flags [none], proto unknown (115), length 74)
172.16.1.253 > 172.16.2.253: ip-proto-115 54
1:48:28.416083 IP (tos 0x0, ttl 62, id 18040, offset 0, flags [none], proto unknown (115), length 74)
172.16.2.253 > 172.16.1.253: ip-proto-115 54
1:48:45.877090 IP (tos 0x0, ttl 64, id 44998, offset 0, flags [none], proto unknown (115), length 105)
172.16.1.253 > 172.16.2.253: ip-proto-115 85

```

FIGURE 8

On a intercepté également les trames échangées entre poste 1 et le routeur 1



lors de l'établissement de la connexion sur l'interface **l2tpeth0** du tunnel grâce à la commande **tcpdump** comme le montre la figure suivante :

```
jean@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns$ sudo ip netns exec routA tcpdump -lnvve -i routA-eth0
tcpdump: listening on routA-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
11:54:45.786449 62:e8:4a:b2:ee:b0 > ea:a8:b7:8b:d8:8a, ethertype IPv4 (0x0800), length 120: (tos 0x0, ttl 6
2, id 595, offset 0, flags [none], proto unknown (115), length 106)
    172.16.2.253 > 172.16.1.253: ip-proto-115 86
11:54:45.787108 ea:a8:b7:8b:d8:8a > 62:e8:4a:b2:ee:b0, ethertype IPv4 (0x0800), length 120: (tos 0x0, ttl 6
4, id 17051, offset 0, flags [none], proto unknown (115), length 106)
    172.16.1.253 > 172.16.2.253: ip-proto-115 86
11:54:45.788079 62:e8:4a:b2:ee:b0 > ea:a8:b7:8b:d8:8a, ethertype IPv4 (0x0800), length 112: (tos 0x0, ttl 6
2, id 596, offset 0, flags [none], proto unknown (115), length 98)
    172.16.2.253 > 172.16.1.253: ip-proto-115 78
11:54:47.567842 62:e8:4a:b2:ee:b0 > ea:a8:b7:8b:d8:8a, ethertype IPv4 (0x0800), length 119: (tos 0x0, ttl 6
2, id 803, offset 0, flags [none], proto unknown (115), length 105)
    172.16.2.253 > 172.16.1.253: ip-proto-115 85
11:54:47.567941 ea:a8:b7:8b:d8:8a > 62:e8:4a:b2:ee:b0, ethertype IPv4 (0x0800), length 112: (tos 0x0, ttl 6
4, id 17191, offset 0, flags [none], proto unknown (115), length 98)
    172.16.1.253 > 172.16.2.253: ip-proto-115 78
```

Les 3 premiers paquets correspondent à l'établissement de la connexion TCP entre les deux postes. Les deux autres paquets correspondent à l'envoi d'un message depuis le poste 1 au routeur 1.

### 4.3 Comparaison du mode d'encapsulation IP et UDP

#### 4.3.1 Encapsulation IP

Pour étudier le fonctionnement du tunnel en mode encapsulation IP, nous avons intercepté le trafic sur l'interface du routeur A à l'aide de la commande **tcpdump** comme le montre la figure suivante :

```
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# tcpdump -lnvv -i routA-eth0tcpdump: listening on routA-eth0
144 bytes
10:21:52.914716 IP (tos 0x0, ttl 64, id 14461, offset 0, flags [none], proto unknown (115), length 74)
    172.16.1.253 > 172.16.2.253: ip-proto-115 54
10:21:52.915125 IP (tos 0x0, ttl 62, id 39300, offset 0, flags [none], proto unknown (115), length 74)
    172.16.2.253 > 172.16.1.253: ip-proto-115 54
10:21:52.915284 IP (tos 0x0, ttl 64, id 14462, offset 0, flags [none], proto unknown (115), length 130)
    172.16.1.253 > 172.16.2.253: ip-proto-115 110
10:21:52.915407 IP (tos 0x0, ttl 62, id 39301, offset 0, flags [none], proto unknown (115), length 130)
    172.16.2.253 > 172.16.1.253: ip-proto-115 110
10:21:53.915824 IP (tos 0x0, ttl 64, id 14502, offset 0, flags [none], proto unknown (115), length 130)
    172.16.1.253 > 172.16.2.253: ip-proto-115 110
10:21:53.916017 IP (tos 0x0, ttl 62, id 39406, offset 0, flags [none], proto unknown (115), length 130)
    172.16.2.253 > 172.16.1.253: ip-proto-115 110
10:21:54.942990 IP (tos 0x0, ttl 64, id 14604, offset 0, flags [none], proto unknown (115), length 130)
    172.16.1.253 > 172.16.2.253: ip-proto-115 110
10:21:54.943193 IP (tos 0x0, ttl 62, id 39444, offset 0, flags [none], proto unknown (115), length 130)
    172.16.2.253 > 172.16.1.253: ip-proto-115 110
```

On remarque que le trafic ICMP sniffé est encapsulé dans un datagramme IP où le protocole IP ( proto unknown ) n'est pas visible lorsqu'on sniffe les trames passant par le routeur A.

#### 4.3.2 Encapsulation UDP

Contrairement à une encapsulation IP, le fonctionnement du tunnel en mode UDP permet de montrer le protocole UDP ( proto UDP ) dans la datagramme IP qui est encapsulé dans les trames comme le montre la figure suivante.

On a capturé le trafic du tunnel sur le routeur A pendant un ping du poste 3 vers le poste 1 :

```
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# tcpdump -lnvv -i routA-eth0
tcpdump: listening on routA-eth0, link-type EN10MB (Ethernet), capture size 2621
44 bytes
10:10:55.743829 IP (tos 0x0, ttl 64, id 25688, offset 0, flags [none], proto UDP
(17), length 114)
    172.16.1.253.5011 > 172.16.2.253.5010: [no cksum] UDP, length 86
10:10:55.991874 IP (tos 0x0, ttl 64, id 25747, offset 0, flags [none], proto UDP
(17), length 142)
    172.16.1.253.5011 > 172.16.2.253.5010: [no cksum] UDP, length 114
10:10:55.992779 IP (tos 0x0, ttl 62, id 19887, offset 0, flags [none], proto UDP
(17), length 142)
    172.16.2.253.5010 > 172.16.1.253.5011: [no cksum] UDP, length 114
10:10:56.992914 IP (tos 0x0, ttl 64, id 25879, offset 0, flags [none], proto UDP
(17), length 142)
    172.16.1.253.5011 > 172.16.2.253.5010: [no cksum] UDP, length 114
10:10:56.993099 IP (tos 0x0, ttl 62, id 20137, offset 0, flags [none], proto UDP
(17), length 142)
    172.16.2.253.5010 > 172.16.1.253.5011: [no cksum] UDP, length 114
```

Enfin, voici une comparaison de la MTU et de la bande passante pour une encapsulation IP dans un premier temps puis UDP.

```
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns/scriptsTunnel# iperf -m -c 192.168.100.254
-----
Client connecting to 192.168.100.254, TCP port 5001
TCP window size: 85.0 KByte (default)
-----
[ 3] local 192.168.100.253 port 55044 connected with 192.168.100.254 port 5001
[ ID] Interval      Transfer      Bandwidth
[ 3] 0.0-10.0 sec  1.18 GBytes  1.02 Gbits/sec
[ 3] MSS size 1406 bytes (MTU 1446 bytes, unknown interface)
```

**FIGURE 9** – MTU et bande passante pour une encapsulation IP



```
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# sudo iperf -m -c 192.168.100.254
-----
Client connecting to 192.168.100.254, TCP port 5001
TCP window size: 45.0 KByte (default)
-----
[ 3] local 192.168.100.253 port 55034 connected with 192.168.100.254 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.0 sec  1.15 GBytes  990 Mbits/sec
[ 3] MSS size 1394 bytes (MTU 1434 bytes, unknown interface)
```

**FIGURE 10** – MTU et bande passante pour une encapsulation UDP

On remarque que l'encapsulation UDP réduit légèrement la MTU.

#### 4.4 Mise en place d'un tunnel GRE en mode GRE-TAP

La mise en place du tunnel GRE en mode GRE-TAP est assez simple puisqu'une partie du travail est déjà faite lors de la configuration du tunnel L2TPv3 et du bridge. On saisit seulement, en plus, les commandes suivantes :

```
#Ajout du tunnel GRE
#Cote R1
ip netns exec rout1 ip link add eoip1 type gretap remote 172.16.2.253 local 172.16.1.253 nopmtudisc
ip netns exec rout1 brctl addif tunnel eoip1
#Cote R2
ip netns exec rout2 ip link add eoip1 type gretap remote 172.16.1.253 local 172.16.2.253 nopmtudisc
ip netns exec rout2 brctl addif tunnel eoip1
```

**FIGURE 11** – Commandes à saisir pour configurer le tunnel GRE

Par défaut, c'est le tunnel L2TPv3 qui est utilisé, on n'active donc pas l'interface du tunnel GRE tout de suite. Pas besoin donc de saisir "ip netns exec rout1 ip link set eoip1 up". A partir de là, le tunnel est prêt à être utilisé. Il suffit de désactiver l'interface du tunnel L2TPv3 et d'activer l'interface du tunnel GRE.

**L'utilisateur peut facilement passer d'un tunnel GRE à L2TPv3 grâce au script "switch-tunnel" :**

```
#!/bin/bash
#Switch entre le tunnel GRE et L2TPv3
var=$(sudo ip netns exec rout1 cat /sys/class/net/eoip1/operstate)
if [ "$var" = "down" ];then
    echo -e 'Desactivation du tunnel l2tpeth0 et activation du tunnel GRE \n'
    sudo ip netns exec rout1 ip link set dev l2tpeth0 down
    sudo ip netns exec rout1 ip link set dev eoip1 up

    sudo ip netns exec rout2 ip link set dev l2tpeth0 down
    sudo ip netns exec rout2 ip link set dev eoip1 up
else
    echo -e 'Desactivation du tunnel GRE et activation du tunnel l2tpeth0 \n'
    sudo ip netns exec rout1 ip link set dev eoip1 down
    sudo ip netns exec rout1 ip link set dev l2tpeth0 up

    sudo ip netns exec rout2 ip link set dev eoip1 down
    sudo ip netns exec rout2 ip link set dev l2tpeth0 up
fi
```

**FIGURE 12** – Script permettant de passer d'un mode à l'autre pour le tunnel (GRE/L2TPv3)

En fonction de l'interface active et donc du mode du tunnel, le script désactive une interface et réactive l'autre. Voici un exemple d'exécution où l'on capture du trafic dans le tunnel pendant que l'on exécute le script "switch\_tunnel"

```

    IP (tos 0x0, ttl 64, id 20621, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.100.2 > 192.168.100.1: ICMP echo request, id 17509, seq 125, length 64
22:09:31.295179 IP (tos 0x0, ttl 62, id 1109, offset 0, flags [none], proto GRE (47), length 126)
    172.16.2.253 > 172.16.1.253: GREv0, Flags [none], length 106
    IP (tos 0x0, ttl 64, id 64418, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.100.1 > 192.168.100.2: ICMP echo reply, id 17509, seq 125, length 64
22:09:32.318733 IP (tos 0x0, ttl 64, id 27140, offset 0, flags [none], proto GRE (47), length 126)
    172.16.1.253 > 172.16.2.253: GREv0, Flags [none], length 106
    IP (tos 0x0, ttl 64, id 20877, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.100.2 > 192.168.100.1: ICMP echo request, id 17509, seq 126, length 64
22:09:32.318882 IP (tos 0x0, ttl 62, id 1294, offset 0, flags [none], proto GRE (47), length 126)
    172.16.2.253 > 172.16.1.253: GREv0, Flags [none], length 106
    IP (tos 0x0, ttl 64, id 64580, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.100.1 > 192.168.100.2: ICMP echo reply, id 17509, seq 126, length 64
22:09:32.354808 IP (tos 0x0, ttl 64, id 38925, offset 0, flags [none], proto unknown (115), length 118)
    172.16.1.253 > 172.16.2.253: ip-proto-115 98
22:09:32.374743 IP (tos 0x0, ttl 62, id 40069, offset 0, flags [none], proto unknown (115), length 118)
    172.16.2.253 > 172.16.1.253: ip-proto-115 98
22:09:32.498890 IP (tos 0x0, ttl 64, id 38958, offset 0, flags [none], proto unknown (115), length 114)
    172.16.1.253 > 172.16.2.253: ip-proto-115 94
22:09:32.538842 IP (tos 0x0, ttl 62, id 40105, offset 0, flags [none], proto unknown (115), length 114)
    172.16.2.253 > 172.16.1.253: ip-proto-115 94
22:09:32.638879 IP (tos 0x0, ttl 64, id 38959, offset 0, flags [none], proto unknown (115), length 118)
    172.16.1.253 > 172.16.2.253: ip-proto-115 98

jean@jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns
Fichier Édition Affichage Rechercher Terminal Aide

jean@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns$ sudo ./switch_tunnel
Desactivation du tunnel GRE et activation du tunnel l2tpeth0

```

**FIGURE 13** – Capture du trafic dans le tunnel pendant le changement de mode de ce dernier

Ici, on voit clairement, sur les paquets capturés, le changement de mode du tunnel de GRE à L2TPv3.

## 4.5 Comparaison des tunnels L2TPv3 et GRE

Dans la capture précédente, on peut observer la différence entre des paquets dans le tunnel en mode GRE et L2TPv3. On remarque qu’avec GRE, on peut visualiser le datagramme encapsulé et donc voir les deux machines qui communiquent (ici 192.168.100.1 et 192.168.100.2). Tout d’abord, nous avons testé la bande passante à travers le tunnel en mode GRE et L2tpv3. On constate ici que le tunnel en mode GRE offre une meilleure bande passante que L2TPv3.

```
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# iperf -c 192.168.100.253
-----
Client connecting to 192.168.100.253, TCP port 5001
TCP window size: 85.0 KByte (default)
-----
[ 3] local 192.168.100.2 port 60908 connected with 192.168.100.253 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3]  0.0-10.0 sec  818 MBytes  686 Mbits/sec
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# ./switch_tunnel
Desactivation du tunnel l2tpeth0 et activation du tunnel GRE

root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# iperf -c 192.168.100.253
-----
Client connecting to 192.168.100.253, TCP port 5001
TCP window size: 85.0 KByte (default)
-----
[ 3] local 192.168.100.2 port 60928 connected with 192.168.100.253 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3]  0.0-10.0 sec  2.23 GBytes  1.91 Gbits/sec
```

**FIGURE 14** – Bande passante dans le tunnel utilisant L2TPv3 dans un premier temps puis GRE

Dans le cas d'un tunnel GRE la MTU est de 1476. (1500 octets moins 24 octets). Cela est logique puisque 20 octets sont nécessaires pour l'entête IP encapsulée ainsi que 4 octets pour l'entête GRE. Dans le cas d'un tunnel L2TPv3 la MTU est 1458.

## 5 Mise en place de chiffrement avec IPsec sur le tunnel

Pour la mise en place du chiffrement avec IPsec, nous avons utilisé la commande `ip xfrm` comme indiqué en utilisant un AH (Authentication Header) et un ESP. Voici les commandes à saisir sur le routeur 1 par exemple :

```
# Cote rout1
ip netns exec rout1 ip xfrm state flush
ip netns exec rout1 ip xfrm policy flush
# AH, " Authentication Header ", pour l'authentification utilisant un clé de longueur 256 bit et ESP, " Encapsulating Security Payload ", pour le chiffrement et
l'authentification " utilisant une clé de longueur 160 bit

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
0x323730ed6f1b9ff0cb084af15b197e862b7c18424a7cdfb74cd385ae23bc4f17 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

# Les politiques de sécurité
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
```

FIGURE 15 – Commandes à réaliser sur le routeur 1

Voici, ci-dessous, un exemple où l'on ping le poste 3 depuis le poste 1, où l'on capture le trafic sur l'interface 0 du routeur 1 et où du chiffrement a été mis en place avec IPsec entre le routeur 1 et le routeur2 :

The image shows two terminal windows. The top window, titled 'root@jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns', shows the execution of 'tcpdump' on the 'rout1-eth0' interface. It captures three packets: two ESP (Encapsulating Security Payload) packets from 172.16.2.253 to 172.16.1.253 and one from 172.16.1.253 to 172.16.2.253. The bottom window, titled 'jean@jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns', shows a 'ping' command from 'poste1' to '192.168.100.2'. The output shows a successful ping with 64 bytes of data, an ICMP sequence of 1, a TTL of 64, and a time of 1.42 ms. Ping statistics show 1 packet transmitted, 1 received, and 0% packet loss.

```
root@jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns
Fichier Édition Affichage Rechercher Terminal Aide
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# sudo ip netns exec rout1 tcpdump ip -lnvv -i rout1-eth0
tcpdump: listening on rout1-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:17:56.163693 IP (tos 0x0, ttl 62, id 16912, offset 0, flags [none], proto ESP (50), length 160)
172.16.2.253 > 172.16.1.253: ESP(spi=0x12345678,seq=0x8f), length 140
17:17:56.164282 IP (tos 0x0, ttl 64, id 65175, offset 0, flags [none], proto ESP (50), length 160)
172.16.1.253 > 172.16.2.253: ESP(spi=0x12345678,seq=0x8f), length 140
17:17:56.512478 IP (tos 0x0, ttl 64, id 65175, offset 0, flags [none], proto ESP (50), length 140)
172.16.1.253 > 172.16.2.253: ESP(spi=0x12345678,seq=0x8f), length 140

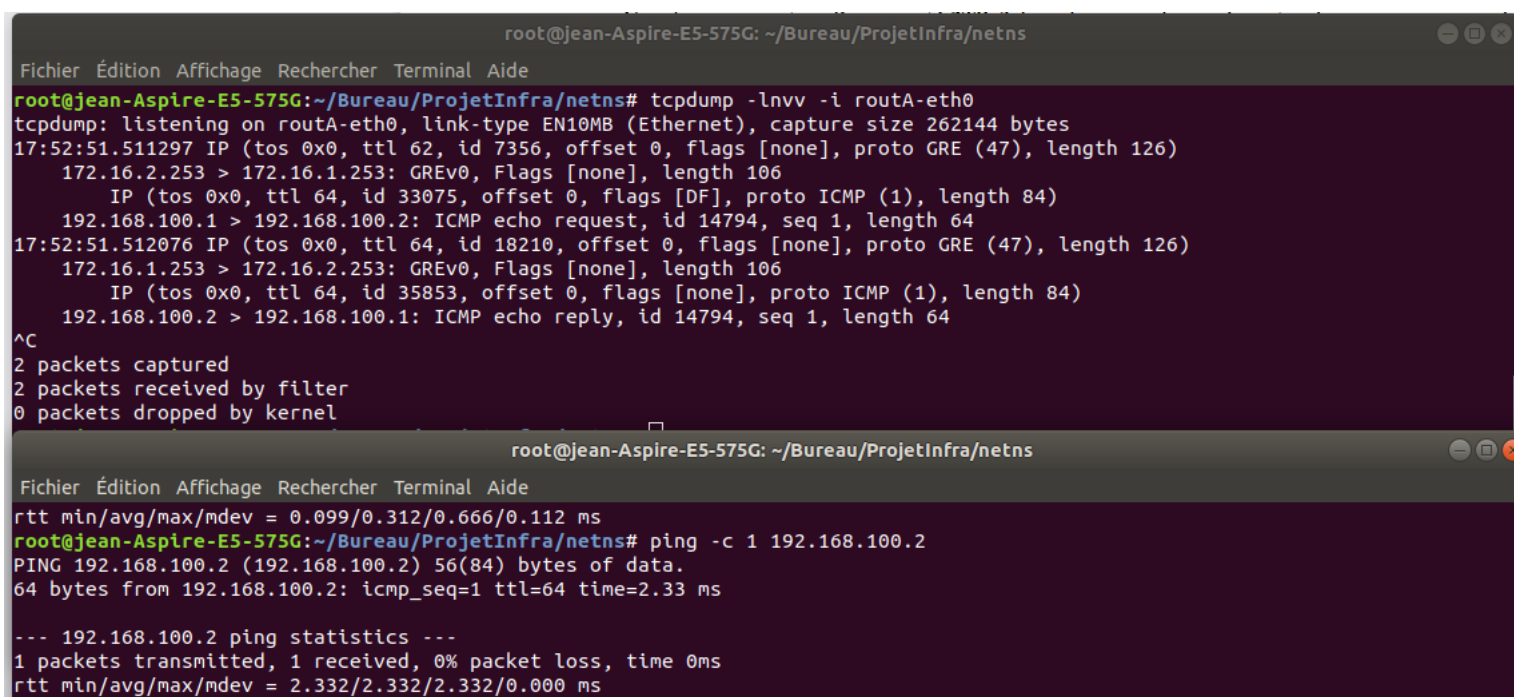
jean@jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns
Fichier Édition Affichage Rechercher Terminal Aide
jean@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns$ sudo ip netns exec poste1 ping -c 1 192.168.100.2
PING 192.168.100.2 (192.168.100.2) 56(84) bytes of data.
64 bytes from 192.168.100.2: icmp_seq=1 ttl=64 time=1.42 ms

--- 192.168.100.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.425/1.425/1.425/0.000 ms
jean@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns$
```

FIGURE 16 – Capture de paquets chiffrés avec IPsec

La différence est encore plus visible en utilisant le tunnel GRE mis en place précédemment. Voici une capture de paquets sur le routeur A dans le cas où,

d'une part, du chiffrement a été mis en place au niveau du tunnel et, d'autre part, dans le cas où aucun chiffrement n'a été mis en place au niveau du tunnel. Dans le cas sans chiffrement, tcpdump nous affiche que le protocole utilisé dans l'entête IP est GREv0 et surtout il affiche le datagramme IP encapsulé nous permettant de voir les machines communicants. (Ici 192.168.100.2 et 192.168.100.1). Dans le cas où du chiffrement a été mis en place, le datagramme IP encapsulé est totalement invisible.



```

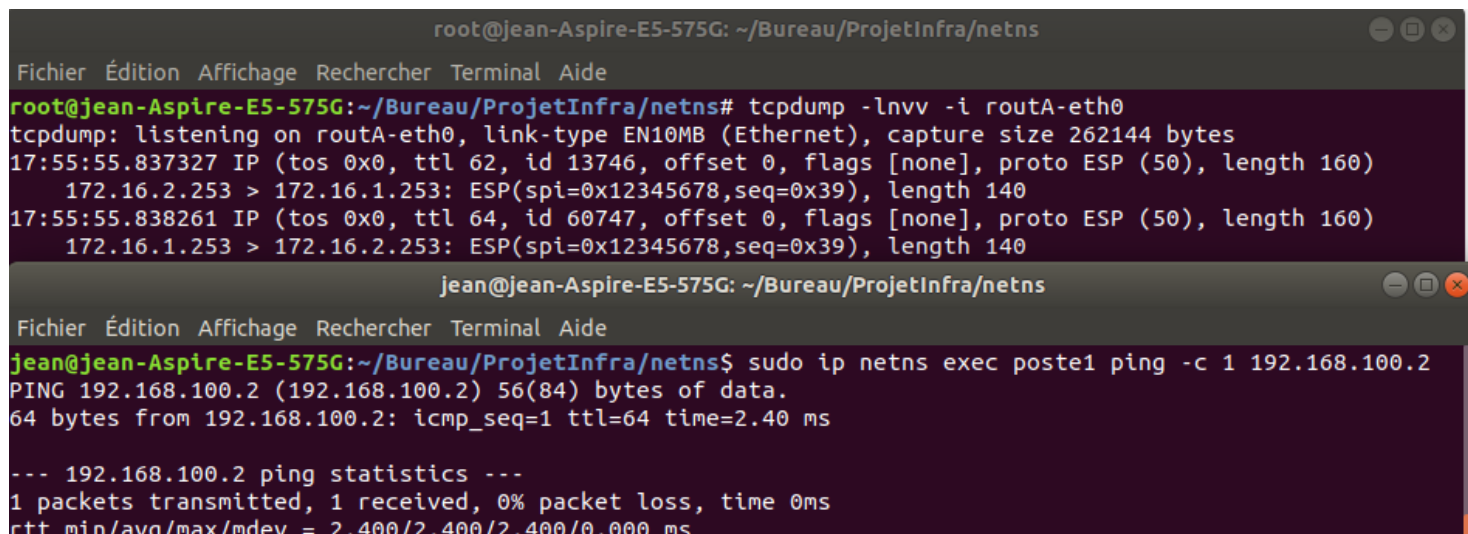
root@jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns
Fichier Édition Affichage Rechercher Terminal Aide
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# tcpdump -lnvv -i routA-eth0
tcpdump: listening on routA-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:52:51.511297 IP (tos 0x0, ttl 62, id 7356, offset 0, flags [none], proto GRE (47), length 126)
  172.16.2.253 > 172.16.1.253: GREv0, Flags [none], length 106
    IP (tos 0x0, ttl 64, id 33075, offset 0, flags [DF], proto ICMP (1), length 84)
      192.168.100.1 > 192.168.100.2: ICMP echo request, id 14794, seq 1, length 64
17:52:51.512076 IP (tos 0x0, ttl 64, id 18210, offset 0, flags [none], proto GRE (47), length 126)
  172.16.1.253 > 172.16.2.253: GREv0, Flags [none], length 106
    IP (tos 0x0, ttl 64, id 35853, offset 0, flags [none], proto ICMP (1), length 84)
      192.168.100.2 > 192.168.100.1: ICMP echo reply, id 14794, seq 1, length 64
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel

root@jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns
Fichier Édition Affichage Rechercher Terminal Aide
rtt min/avg/max/mdev = 0.099/0.312/0.666/0.112 ms
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# ping -c 1 192.168.100.2
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=2.33 ms

--- 192.168.100.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.332/2.332/2.332/0.000 ms

```

**FIGURE 17** – Capture d'un paquet **non chiffré** passant par le tunnel GRE



```

root@jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# tcpdump -lnvv -i routA-eth0
tcpdump: listening on routA-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:55:55.837327 IP (tos 0x0, ttl 62, id 13746, offset 0, flags [none], proto ESP (50), length 160)
 172.16.2.253 > 172.16.1.253: ESP(spi=0x12345678,seq=0x39), length 140
17:55:55.838261 IP (tos 0x0, ttl 64, id 60747, offset 0, flags [none], proto ESP (50), length 160)
 172.16.1.253 > 172.16.2.253: ESP(spi=0x12345678,seq=0x39), length 140

jean@jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
jean@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns$ sudo ip netns exec poste1 ping -c 1 192.168.100.2
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=2.40 ms

--- 192.168.100.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.400/2.400/2.400/0.000 ms

```

**FIGURE 18** – Capture d'un paquet **chiffré avec IPsec** et passant par le tunnel GRE

Dans le cas où l'on utilise du chiffrement, on peut notamment voir la SPI «Security Parameters Index» utilisée pour le chiffrement des paquets. De plus, si l'on avait spécifié "icmp" dans la commande tcpdump alors le routeur 1 n'aurait rien capturé puisque tcpdump ne peut pas identifier le paquet chiffré comme étant un "ping" (puisque le paquet est chiffré justement). Cependant, le chiffrement permet notamment de cacher l'adresse des machines communicant à travers le tunnel ainsi que les données envoyées.

Le point faible du chiffrement des paquets avec IPsec et que, fatalement, il diminue le débit. A l'aide d'iperf, nous avons mesuré le débit maximal entre le routeur 1 et le poste 1 avec et sans chiffrement. Pour cela, on lance sur le routeur 1 la commande "**iperf -V -s**". Le routeur 1 est donc en attente d'une connexion TCP sur le port 5001. De l'autre côté, il faut se connecter sur le routeur 1 sur le port 5001 depuis le poste 1 : "**iperf -c 192.168.100.254**"



```
4 packets transmitted, 4 received, 0% packet loss, time 3038ms
rtt min/avg/max/mdev = 0.354/1.111/3.163/1.185 ms
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# 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 192.168.100.1 port 58956 connected with 192.168.100.254 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.0 sec   365 MBytes  306 Mbits/sec
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns#
```

FIGURE 19 – Débit mesuré avec iperf **avec** du chiffrement avec IPsec

```
rtt min/avg/max/mdev = 0.278/0.292/0.319/0.027 ms
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# 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 192.168.100.1 port 58980 connected with 192.168.100.254 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.0 sec   998 MBytes  836 Mbits/sec
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns#
```

FIGURE 20 – Débit mesuré avec iperf **sans** chiffrement

L'utilisation de chiffrement avec IPsec diminue le débit moyen de 836MB/s à 305MB/s ce qui n'est pas négligeable. Le chiffrement des paquets avec IPsec impacte donc grandement le débit. Cette diminution du débit s'explique simplement par le temps requis pour chiffrer les paquets. Le débit, lors de l'utilisation du chiffrement avec IPsec, dépend donc aussi de l'algorithme utilisé pour chiffrer les paquets. La MTU et le MSS ("Maximum Segment Size") ne sont cependant pas impactés par le chiffrement ce qui est logique.



## 6 Accès à internet "intelligent"

### 6.1 Accès à internet via le routeur 1

Pour l'accès à internet depuis les postes situés dans les VLANs, nous avons, dans un premier temps, attribué une adresse à la machine physique dans "internet" (le réseau en 10.87.0.0/24). Nous avons ensuite défini du MASQUERADING en sortie de la machine physique de façon à modifier l'adresse source des paquets sortant vers internet. Nous avons également défini une règle de SNAT au niveau du routeur 1 qui est le routeur de sortie donnant accès à internet au VLANs. Tous les paquets sortant sur internet doivent passer par le routeur 1, nous avons donc défini la règle de SNAT à ce niveau-là. Grâce à la règle de SNAT ajoutée sur le routeur 1 pour les paquets sortants, seulement une règle de routage est nécessaire sur la machine physique puisque tous les paquets auront la même adresse source en sortie et donc la même destination (le routeur 1) en entrée : "sudo ip route add 172.16.1.0/24 via 10.87.0.1" où 172.16.1.0/24 est le réseau dans lequel se trouve le routeur 1 qui réalise du SNAT. On adapte ensuite les différentes routes dans les netns notamment les routes par défaut.

```
#Attribuer une adresse dans "internet" à la machine physique
sudo ip a add dev internet 10.87.0.3/24

#On modifie la route par défaut sur routeA vers internet
sudo ip netns exec routA ip route del default via 10.87.0.2
sudo ip netns exec routA ip route add default via 10.87.0.3

#modifie l'adresse source des paquets en sortie sur rou1-eth0
sudo ip netns exec rout1 iptables -t nat -o rout1-eth0 -A POSTROUTING -j MASQUERADE

#Puisqu on fait du SNAT on a besoin de rajouter seulement une seule regle de routage sur
sudo ip route add 172.16.1.0/24 via 10.87.0.1

#On fait du MASQUERADING pour pouvoir avoir une reponse d'internet
sudo iptables -t nat -A POSTROUTING -o enp4s0f1 -j MASQUERADE
sudo iptables -t nat -A POSTROUTING -o wlp3s0 -j MASQUERADE

#On active la fonction de routage sur la VM/machine Linux
sudo sysctl -w net.ipv4.conf.all.forwarding=1
sudo sysctl -w net.ipv4.conf.all.rp_filter=0

#On ajoute une route par défaut sur les postes vers la sortie
sudo ip netns exec poste1 ip route add default via 192.168.100.253
sudo ip netns exec poste2 ip route add default via 192.168.200.253
sudo ip netns exec poste3 ip route add default via 192.168.100.254
sudo ip netns exec poste4 ip route add default via 192.168.200.254
```

**FIGURE 21** – Première partie des commandes permettant de donner accès à internet

A partir de là, seulement les postes 3 et 4 ont accès à internet. Il faut désormais définir les règles de routage de façon à ce que le poste 1 et le poste 2 aient également accès à internet **via le routeur 1** en empruntant chacun leurs tunnels respectifs (tunnel.100 et tunnel.200). De la routing policy est donc nécessaire pour router le paquet en fonction de sa source et non de sa destination.

```
#Le trafic du lan 100 à destination d'internet passe par le tunnel.100 pour atteindre R1 sur
#Le trafic du lan 200 à destination d'internet passe par le tunnel.200 pour atteindre R1 sur
sudo ip netns exec rout2 ip rule add from 192.168.100.0/24 lookup versTnl100
sudo ip netns exec rout2 ip rule add from 192.168.200.0/24 lookup versTnl200

#On remplit les tables de routages
sudo ip netns exec rout2 ip route add default via 192.168.100.254 table versTnl100
sudo ip netns exec rout2 ip route add 192.168.100.0/24 dev tunnel.100 table versTnl100

sudo ip netns exec rout2 ip route add default via 192.168.200.254 table versTnl200
sudo ip netns exec rout2 ip route add 192.168.200.0/24 dev tunnel.200 table versTnl200
```

**FIGURE 22** – Deuxième partie des commandes permettant de donner accès à internet

Les paquets provenant d'une machine dans la VLAN 100 sont envoyés par tunnel 100 tandis que les paquets provenant d'une machine dans la VLAN 200 sont envoyés dans le tunnel 200. Il ne faut, bien sûr, pas oublier d'ajouter des règles de routages à ces tables. A partir de là, tous les paquets provenant des VLANs et à destination sont envoyés sur le routeur 1 qui lui-même les envoie vers 10.87.0.0.3 (la machine physique) après avoir réalisé du SNAT.

The image shows three terminal windows. The top-left window shows a tcpdump capture on tunnel.100, displaying ICMP echo requests and replies between 192.168.100.1 and 8.8.8.8. The top-right window shows a tcpdump capture on rout1-eth0, displaying ICMP echo requests and replies between 172.16.1.253 and 8.8.8.8. The bottom window shows ping statistics for 8.8.8.8 from the perspective of the netns, indicating 0% packet loss and a time of 11.4 ms.

```
Fichier Édition Affichage Rechercher Terminal Aide
root@jean-Aspire-E5-575G:~# sudo tcpdump icmp -lnvv -i tunnel.100
tcpdump: listening on tunnel.100, link-type EN10MB (Ethernet), capture size 2621
44 bytes
22:13:36.951675 IP (tos 0x0, ttl 63, id 44821, offset 0, flags [DF], proto ICMP
(1), length 84)
192.168.100.1 > 8.8.8.8: ICMP echo request, id 24917, seq 1, length 64
22:13:36.962116 IP (tos 0x0, ttl 113, id 0, offset 0, flags [none], proto ICMP (
1), length 84)
8.8.8.8 > 192.168.100.1: ICMP echo reply, id 24917, seq 1, length 64

Fichier Édition Affichage Rechercher Terminal Aide
root@jean-Aspire-E5-575G:~# sudo tcpdump icmp -lnvv -i rout1-eth0
tcpdump: listening on rout1-eth0, link-type EN10MB (Ethernet), capture size 2621
44 bytes
22:13:36.951738 IP (tos 0x0, ttl 62, id 44821, offset 0, flags [DF], proto ICMP
(1), length 84)
172.16.1.253 > 8.8.8.8: ICMP echo request, id 24917, seq 1, length 64
22:13:36.962102 IP (tos 0x0, ttl 114, id 0, offset 0, flags [none], proto ICMP (
1), length 84)
8.8.8.8 > 172.16.1.253: ICMP echo reply, id 24917, seq 1, length 64

Jean@Jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns
Fichier Édition Affichage Rechercher Terminal Aide
--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 12.084/12.084/12.084/0.000 ms
Jean@Jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns$ sudo ip netns exec poste1 ping -c 1 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
From 192.168.100.253: icmp_seq=1 Redirect Host(New nexthop: 192.168.100.254)
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=11.4 ms
```

**FIGURE 23** – Vérification que le trafic à destination d'internet du poste 1 passe bien par le routeur 1

La capture ci-contre, nous montre un ping de 8.8.8.8 depuis le poste1 ainsi qu'une capture de paquet sur l'interface tunnel et l'interface rout1-eth0 sur le routeur 1. Si on observe les paquets capturés, on remarque bien que le paquet

passer par le tunnel.100 pour atteindre le routeur 1. On remarque, de plus, que le SNAT est bien mis en place puisque le paquet sortant du routeur 1 et partant sur internet a bien l'adresse IP source qui a été modifiée par 172.16.1.253 (l'adresse du routeur 1). Enfin sur le poste 1 on obtient bien une réponse de la machine pingée.

Une autre méthode plus simple, permettant par la suite de rediriger les postes 1 et 2 sur le routeur, est de directement modifier les routes par défaut de poste1 et poste 2 pour les diriger directement vers le routeur 1 :

Dans le script, on modifie donc les routes comme tel :

```
sudo ip netns exec poste1 ip route add default via 192.168.100.254
sudo ip netns exec poste2 ip route add default via 192.168.200.254
sudo ip netns exec poste3 ip route add default via 192.168.100.254
sudo ip netns exec poste4 ip route add default via 192.168.200.254
```

Par défaut, le trafic du poste 1 et 2 part directement sur le routeur 1 par les tunnels. Les règles de "Policy routing ne sont donc pas nécessaires avec ces règles de routage". Voici donc la deuxième version du script donnant l'accès à internet par le Routeur1 et préparant l'accès à internet par le routeur 2 en anticipation de la question suivante.

```
#Attribuer une adresse dans "internet" à la machine physique
sudo ip a add dev internet 10.87.0.3/24

#On modifie la route par défaut sur routeA vers internet
sudo ip netns exec routA ip route del default via 10.87.0.2
sudo ip netns exec routA ip route add default via 10.87.0.3

#modifie l'adresse source des paquets en sortie sur rout1-eth0
sudo ip netns exec rout1 iptables -t nat -o rout1-eth0 -A POSTROUTING -j MASQUERADE
#En prevision de la Q2 avec dnsmasq
sudo ip netns exec rout2 iptables -t nat -o rout2-eth0 -A POSTROUTING -j MASQUERADE

#Puisqu on fait du SNAT on a besoin de rajouter seulement une seule regle de routage sur
sudo ip route add 172.16.1.0/24 via 10.87.0.1
#En prevision de quand rout2 sera le routeur de sortie pour poste 1 et 2
sudo ip route add 172.16.2.0/24 via 10.87.0.2

#On fait du MASQUERADING pour pouvoir avoir une reponse d'internet
sudo iptables -t nat -A POSTROUTING -o enp4s0f1 -j MASQUERADE
sudo iptables -t nat -A POSTROUTING -o wlp3s0 -j MASQUERADE

#On active la fonction de routage sur la VM/machine Linux
sudo sysctl -w net.ipv4.conf.all.forwarding=1
sudo sysctl -w net.ipv4.conf.all.rp_filter=0

#On ajoute une route par défaut sur les postes vers la sortie
sudo ip netns exec poste1 ip route add default via 192.168.100.254
sudo ip netns exec poste2 ip route add default via 192.168.200.254
sudo ip netns exec poste3 ip route add default via 192.168.100.254
sudo ip netns exec poste4 ip route add default via 192.168.200.254
```

FIGURE 24 – Version simplifiée du script donnant accès à internet

## 6.2 Redirection avec dnsmasq des postes 1 et 2 vers le routeur 2 pour optimiser l'accès

Nous avons vu, dans le script de la question précédente que tous les postes possèdent une route par défaut vers le routeur 1. L'objectif dans cette question va être de modifier la route par défaut de poste 1 et 2 lorsqu'il obtient une adresse par DHCP. Il va donc falloir que la route par défaut soit le routeur 2. Dans la question précédente, nous avons déjà ajouté la route vers le routeur 2 sur la machine physique permettant le retour des paquets et le SNAT sur le routeur 2 est également déjà en place. Il ne manque donc plus qu'à modifier la commande dnsmasq du côté du serveur pour qu'il indique au poste se configurant par DHCP d'emprunter une certaine route par défaut. Pour cela on rajoute simplement l'option : **-dhcp-option=option :router,192.168.200.253** et **-dhcp-option=option :router,192.168.100.253** à la commande dnsmasq sai-

sie précédemment. Les adresses indiquées sont les adresses des interfaces des tunnels du routeur 2.

On peut observer le changement de la route par défaut du poste1 après sa configuration par DHCP. La route par défaut n'est plus le routeur 1 mais routeur 2 désormais :

```
jean@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns$ sudo ip netns exec poste1 ip route
default via 192.168.100.254 dev poste1-eth0.100
192.168.100.0/24 dev poste1-eth0.100 proto kernel scope link src 192.168.100.1
jean@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns$ sudo ./scriptsDHCP/AutoConfig_Poste1_DHCP &> out.log
jean@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns$ sudo ip netns exec poste1 ip route
default via 192.168.100.253 dev poste1-eth0.100
192.168.100.0/24 dev poste1-eth0.100 proto kernel scope link src 192.168.100.33
jean@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns$
```

**FIGURE 25** – Changement de route par défaut sur le poste 1

On peut, enfin, vérifier que le trafic passe bien par le routeur 2 avant de sortir sur internet sans passer par le routeur 1 :

```
jean@jean-Aspire-E5-575G: ~
Fichier Édition Affichage Rechercher Terminal Aide
jean@jean-Aspire-E5-575G:~$ sudo ip netns exec rout1 tcpdump icmp -lnvv -i rout1-eth0
tcpdump: listening on rout1-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
44 bytes
[ ]

jean@jean-Aspire-E5-575G: ~
Fichier Édition Affichage Rechercher Terminal Aide
jean@jean-Aspire-E5-575G:~$ clear
jean@jean-Aspire-E5-575G:~$ sudo tcpdump icmp -lnvv -i internet
tcpdump: listening on internet, link-type EN10MB (Ethernet), capture size 262144 bytes
11:31:56.749369 IP (tos 0x0, ttl 61, id 32566, offset 0, flags [DF], proto ICMP (1), length 84)
172.16.2.253 > 8.8.8.8: ICMP echo request, id 27653, seq 1, length 64
11:31:56.759450 IP (tos 0x0, ttl 115, id 0, offset 0, flags [none], proto ICMP (1), length 84)
8.8.8.8 > 172.16.2.253: ICMP echo reply, id 27653, seq 1, length 64
[ ]

jean@jean-Aspire-E5-575G: ~
Fichier Édition Affichage Rechercher Terminal Aide
jean@jean-Aspire-E5-575G:~$ clear
jean@jean-Aspire-E5-575G:~$ sudo ip netns exec route2 tcpdump icmp -lnvv -i route2-eth0
tcpdump: listening on route2-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
11:31:56.748710 IP (tos 0x0, ttl 63, id 32566, offset 0, flags [DF], proto ICMP (1), length 84)
172.16.2.253 > 8.8.8.8: ICMP echo request, id 27653, seq 1, length 64
11:31:56.760267 IP (tos 0x0, ttl 114, id 0, offset 0, flags [none], proto ICMP (1), length 84)
8.8.8.8 > 172.16.2.253: ICMP echo reply, id 27653, seq 1, length 64
[ ]

root@jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns
Fichier Édition Affichage Rechercher Terminal Aide
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns$ sudo ip netns exec poste1 ping -c 1 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=12.2 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 12.294/12.294/12.294/0.000 ms
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns$
```

**FIGURE 26** – Vérification du changement de route par défaut

On réalise un ping vers internet depuis le poste 1 configuré par DHCP et dont la route par défaut a été modifié. On observe bien que le trafic passe par le routeur2 avant de partir directement sur la machine physique sans passer par le routeur 1. De plus, on constate que le routeur 2 a bien réalisé le SNAT puisque l'adresse source a été modifié par l'adresse du routeur 2.

## 7 Blocage des VLANs

### 7.1 A l'aide du firewall

Le blocage des VLANs, à partir de règles de firewall, peut être réalisé de nombreuses façons différentes. Notamment en utilisant une policy DROP et en acceptant seulement les paquets allant vers le même VLAN ou internet. Une autre manière, plus simple, est de rejeter les paquets ayant en source une VLAN et en destination une VLAN différente. Cette action doit être réalisée sur le routeur 1 et 2.

- `iptables -t filter -A FORWARD -s 192.168.100.0/24 -d 192.168.200.0/24 -j REJECT`
- `iptables -t filter -A FORWARD -s 192.168.200.0/24 -d 192.168.100.0/24 -j REJECT`

Cependant, une seule des deux règles ci-dessus est indispensable. En effet, avec même avec une de ces deux règles la communication entre le VLAN 100 et 200 ne pourra pas se faire. Si on saisit seulement la première règle, par exemple, une machine dans le VLAN 100 ne pourra pas atteindre la VLAN 200. Par contre, une machine dans la VLAN 200 pourra quand même atteindre une machine dans la VLAN 100 mais cette dernière ne pourra pas répondre.

Voici un exemple d'exécution où une machine d'une VLAN ping une machine d'une autre VLAN. Le ping fonctionne jusqu'à l'ajout de la règle de firewall.

```

^C
root@jean-Aspire-E5-575G:~# sudo iptables -t filter -A FORWARD -s 192.168.100.0/24
-d 192.168.200.0/24 -j REJECT
root@jean-Aspire-E5-575G:~#
root@jean-Aspire-E5-575G:~# sudo ip netns exec poste3 ping 192.168.200.2
PING 192.168.200.2 (192.168.200.2) 56(84) bytes of data:
64 bytes from 192.168.200.2: icmp_seq=1 ttl=63 time=1.14 ms
64 bytes from 192.168.200.2: icmp_seq=2 ttl=63 time=0.188 ms
64 bytes from 192.168.200.2: icmp_seq=3 ttl=63 time=0.066 ms
64 bytes from 192.168.200.2: icmp_seq=4 ttl=63 time=0.191 ms
From 192.168.100.254 icmp_seq=5 Destination Port Unreachable
From 192.168.100.254 icmp_seq=6 Destination Port Unreachable
From 192.168.100.254 icmp_seq=7 Destination Port Unreachable
From 192.168.100.254 icmp_seq=8 Destination Port Unreachable
From 192.168.100.254 icmp_seq=9 Destination Port Unreachable

```

**FIGURE 27** – Blocage de la communication entre deux machines de deux VLANs différentes

### 7.2 A l'aide de la "Policy Routing"

De nombreux moyens sont possibles pour bloquer le trafic entre deux VLANs avec de la Policy Routing. Une méthode consiste à affecter une table de routage



spécifique à chaque VLAN. Pour cela, il est nécessaire d'identifier la provenance des paquets. Les paquets provenant de 192.168.100.0 sont soumis à une certaine table dans laquelle il n'y a aucune route vers le réseau 192.168.200.0. De même, pour les paquets provenant de 192.168.200.0, ils sont soumis à une autre table spécifique dans laquelle il n'existe aucune route vers le VLAN 192.168.100.0. Le problème de cette méthode est que si aucune règle ne match dans la table, alors il semblerait que le paquet soit soumis alternativement à la table main. Il faut donc supprimer les règles de routages vers les VLANs dans la **table main**.

Voici les commandes à saisir sur le routeur 2, les commandes à saisir sur le routeur 1 sont les sensiblement les mêmes :

```
sudo ip netns exec rout2 ip rule add from 192.168.100.0/24 lookup versTnl100
sudo ip netns exec rout2 ip rule add from 192.168.200.0/24 lookup versTnl200

#On remplit les tables de routages
sudo ip netns exec rout2 ip route add default via 192.168.100.254 table versTnl100
sudo ip netns exec rout2 ip route add 192.168.100.0/24 dev tunnel.100 table versTnl100

sudo ip netns exec rout2 ip route add default via 192.168.200.254 table versTnl200
sudo ip netns exec rout2 ip route add 192.168.200.0/24 dev tunnel.200 table versTnl200
```

**FIGURE 28** – Commandes sur le routeur 2 pour bloquer la communication entre les VLANs

A cela, il est nécessaire de supprimer les routes vers les VLANs dans la table main des deux routeurs :

```
ip route del 192.168.200.0/24 dev tunnel.200
ip route del 192.168.100.0/24 dev tunnel.100
```

Et voici, ci-dessous, l'exécution de la commande ping sur poste1 vers les différents postes des VLANs. On obtient bien le résultat escompté, le poste 1 peut joindre le poste 3 qui est dans le même VLAN mais pas le poste 2 et 4 qui sont dans une VLAN différente.



```

jean@jean-Aspire-E5-575G: ~
Fichier Édition Affichage Rechercher Terminal Aide
jean@jean-Aspire-E5-575G:~$ poste1="192.168.100.1"
jean@jean-Aspire-E5-575G:~$ poste3="192.168.100.2"
jean@jean-Aspire-E5-575G:~$ sudo ip netns exec poste1 ping -c 1 $poste1
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data.
64 bytes from 192.168.100.1: icmp_seq=1 ttl=64 time=0.103 ms
--- 192.168.100.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.103/0.103/0.103/0.000 ms
jean@jean-Aspire-E5-575G:~$ sudo ip netns exec poste1 ping -c 1 $poste3
PING 192.168.100.2 (192.168.100.2) 56(84) bytes of data.
64 bytes from 192.168.100.2: icmp_seq=1 ttl=64 time=0.758 ms
--- 192.168.100.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.758/0.758/0.758/0.000 ms
jean@jean-Aspire-E5-575G:~$

root@jean-Aspire-E5-575G: ~/Bureau/ProjetInfra/netns
Fichier Édition Affichage Rechercher Terminal Aide
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# poste2="192.168.200.1"
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# poste4="192.168.200.2"
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# sudo ip netns exec poste1 ping -c 1 $poste2
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data.
^C
--- 192.168.200.1 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# sudo ip netns exec poste1 ping -c 1 $poste4
PING 192.168.200.2 (192.168.200.2) 56(84) bytes of data.
^C
--- 192.168.200.2 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns#

```

**FIGURE 29** – Blocage de la communication entre deux machines de deux VLANs différentes

Une autre manière plus élégante, est de filtrer les paquets n'ayant pas le même VLAN source et destination avec l'option "prohibit". Voici le script qui permet de réaliser cela :

```

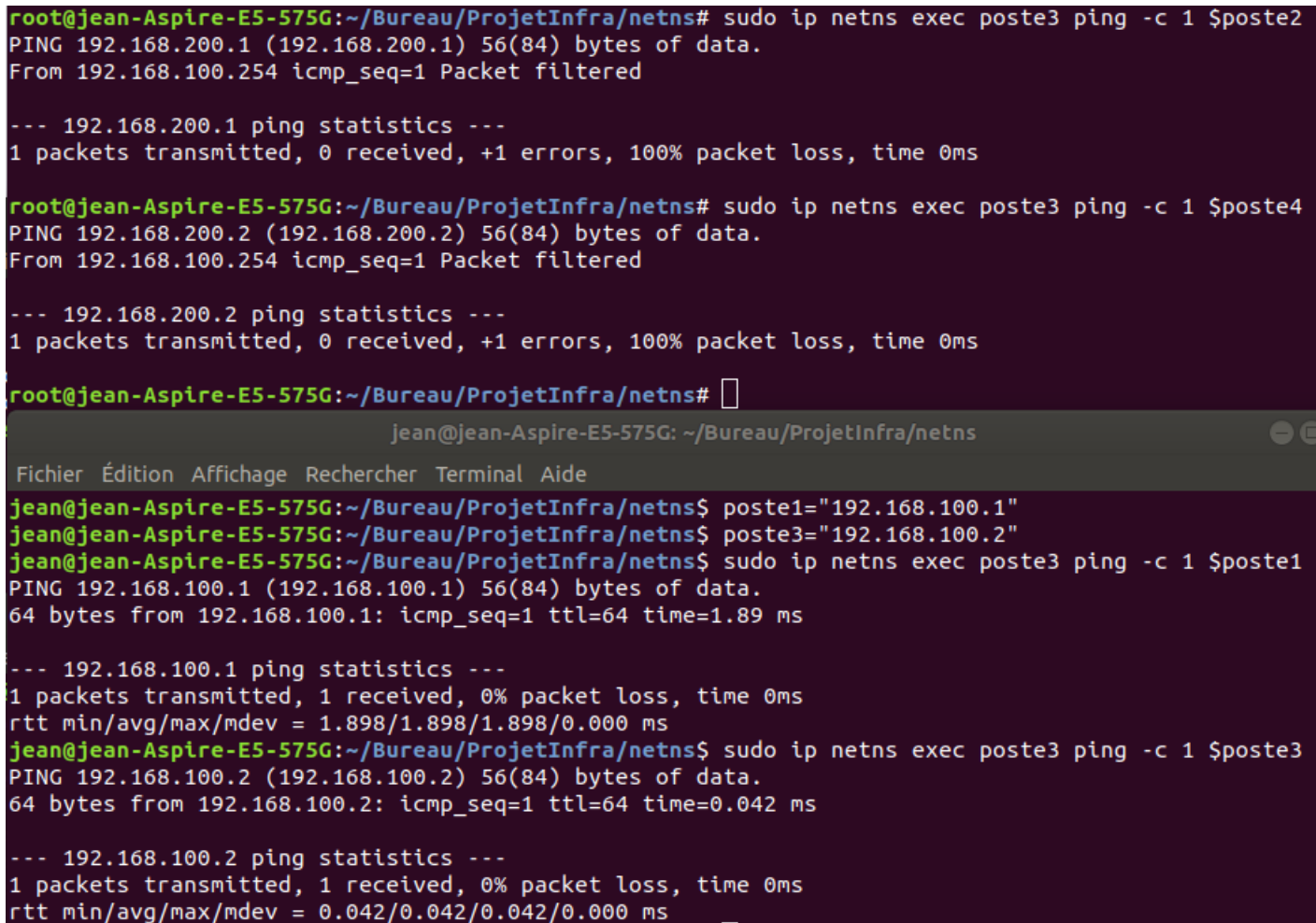
#!/bin/bash -x

#Blocage de la communication entre deux VLANs différentes
sudo ip netns exec rout1 ip rule add from 192.168.100.0/24 to 192.168.200/24 prohibit
sudo ip netns exec rout1 ip rule add from 192.168.200.0/24 to 192.168.100/24 prohibit
sudo ip netns exec rout2 ip rule add from 192.168.200.0/24 to 192.168.100/24 prohibit
sudo ip netns exec rout2 ip rule add from 192.168.100.0/24 to 192.168.200/24 prohibit

```

**FIGURE 30** – Blocage de la communication entre deux machines de deux VLAN différentes

Dans ce cas-là, le prohibit va renvoyer un message d'erreur si une machine d'une VLAN essaie de communiquer avec une machine d'une VLAN différente de la sienne. Ci-dessous, l'exécution de la commande ping sur poste3 vers les différents postes dans les VLANs :



```
root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# sudo ip netns exec poste3 ping -c 1 $poste2
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data.
From 192.168.100.254 icmp_seq=1 Packet filtered

--- 192.168.200.1 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns# sudo ip netns exec poste3 ping -c 1 $poste4
PING 192.168.200.2 (192.168.200.2) 56(84) bytes of data.
From 192.168.100.254 icmp_seq=1 Packet filtered

--- 192.168.200.2 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

root@jean-Aspire-E5-575G:~/Bureau/ProjetInfra/netns#
```

The terminal window shows a user named 'jean' at a machine 'jean-Aspire-E5-575G' in the directory '~/Bureau/ProjetInfra/netns'. The user runs several commands to configure and test network connectivity. First, they set 'poste1' to '192.168.100.1' and 'poste3' to '192.168.100.2'. Then, they run a ping command from 'poste3' to '\$poste1' (192.168.100.1). The output shows that the ping was successful, with 1 packet transmitted and received, 0% packet loss, and a round-trip time of 1.89 ms. Finally, they run a ping command from 'poste3' to '\$poste3' (192.168.100.2). The output shows that the ping was also successful, with 1 packet transmitted and received, 0% packet loss, and a round-trip time of 0.042 ms.

**FIGURE 31** – Blocage de la communication entre deux machines de deux VLAN différentes avec "prohibit"

Cette fois-ci, si une machine essaie de communiquer avec une machine dans un autre VLAN, un message annonçant que le paquet a été filtré s'affiche ("Packet Filtered"). Pour la communication au sein d'un même VLAN cela ne change rien. L'avantage de cette deuxième méthode est qu'il n'est pas nécessaire de modifier les routes sur les routeurs.

## 8 Organisation du projet

Tous les scripts utilisés dans le projet sont disponibles et peuvent être exécutés. Ils doivent cependant être exécutés dans le bon ordre. Par exemple, pour mettre en place le chiffrement du tunnel grâce au script "IPsec", il faut préalablement avoir lancé le script créant le tunnel.

**Pour simplifier les choses nous avons créé un script build\_all permettant de tout construire automatiquement et laissant le choix à l'utilisateur. On lance donc build\_all.**

Après avoir créé l'architecture de base, le script configure les VLANs automatiquement. L'utilisateur doit ensuite choisir entre une encapsulation IP ou UDP pour le tunnel :

```
+ ip netns exec poste2 ip link set dev poste2-eth0.200 up
+ ip netns exec poste3 ip link add link poste3-eth0 name poste3-eth0.100 type vl
an id 100
+ ip netns exec poste3 ip a add dev poste3-eth0.100 192.168.100.2/24
+ ip netns exec poste3 ip link set dev poste3-eth0.100 up
+ ip netns exec poste4 ip link add link poste4-eth0 name poste4-eth0.200 type vl
an id 200
+ ip netns exec poste4 ip a add dev poste4-eth0.200 192.168.200.2/24
+ ip netns exec poste4 ip link set dev poste4-eth0.200 up
+ sudo ip netns exec poste1 ip route add default via 192.168.100.254
+ sudo ip netns exec poste2 ip route add default via 192.168.200.254
+ sudo ip netns exec poste3 ip route add default via 192.168.100.254
+ sudo ip netns exec poste4 ip route add default via 192.168.200.254
Configuration du tunnel L2TPv3 : encapsulation (0)TCP ou (1)UDP : █
```

Le script va ensuite demander à l'utilisateur s'il souhaite que le poste 1 s'auto-configure par DHCP. Si le poste 1 s'auto-configure grâce au DHCP, alors il utilisera le routeur 2 comme routeur de sortie vers internet. Si l'utilisateur choisit que le poste 1 ne s'auto-configure pas par DHCP alors le poste 1 utilisera le routeur 1 comme routeur de sortie vers internet :

```
+ sudo ip netns exec rout1 dnsmasq --interface=tunnel.100 --except-interface=lo
--bind-interfaces --dhcp-range=192.168.100.1,192.168.100.254,255.255.255.0 --dhc
p-option=option:router,192.168.100.253
+ sudo ip netns exec rout1 dnsmasq --interface=tunnel.200 --except-interface=lo
--bind-interfaces --dhcp-range=192.168.200.1,192.168.200.254,255.255.255.0 --dhc
p-option=option:router,192.168.200.253
Voulez-vous reconfigurer poste1 grace au DHCP ? y/n : █
```

Ensuite, le script va demander à l'utilisateur s'il souhaite chiffrer les paquets dans le tunnel grâce à IPsec :

```
p-option=option:router,192.168.100.253
+ sudo ip netns exec rout1 dnsmasq --interface=tunnel.200 --except-interfaces
--bind-interfaces --dhcp-range=192.168.200.1,192.168.200.254,255.255.255.0
p-option=option:router,192.168.200.253
Voulez-vous reconfigurer poste1 grace au DHCP ? y/n :n
Voulez-vous mettre en place un chiffrement IPsec sur le tunnel ? y/n :y
```

Enfin, le script demande à l'utilisateur s'il souhaite bloquer les VLANs grâce à la policy routing vu précédemment :

```
+ ip netns exec rout2 ip xfrm policy add src 172.16.1.253 dst 172.16.2.253 proto esp reqid 0x1
n tmpl src 172.16.1.253 dst 172.16.2.253 proto esp reqid 0x1
rt
Voulez-vous bloquer la communication inter-VLANs ? y/n :y
```

Le script `acces_internet` donnant accès à internet doit être lancé à part. Puisqu'il contient des règles de firewall sur la machine physique, nous avons préféré le séparer du reste. Dans ce script, il est notamment nécessaire de modifier dans les commandes, les noms des interfaces de la machine physique.

Pour finir, voici la liste des scripts utilisés, contenus dans le projet final :

- **build\_architecture** construit l'architecture de départ du projet
- **build\_all** construit l'architecture finale du projet hors mis l'accès à internet qui doit être configuré en exécutant le script "`acces_internet`" et qui contient des règles de routages et firewall sur la machine physique.
- **VLAN** qui configure les VLANs
- **tunnelL2TPv3\_IP**
- **tunnelL2TPv3\_IP**
- **tunnelGRE**
- **DHCPServeur** qui permet de mettre en place le serveur DHCP sur le routeur 1
- **AutoConfig\_Poste1\_DHCP** qui supprime l'adresse du poste 1 et qui reconfigure l'adresse de ce dernier grâce au service DHCP s'exécutant sur le routeur 1
- **IPsec** qui met en place du chiffrement au niveau du tunnel
- **switch\_tunnel** qui permet de basculer d'un mode à l'autre pour le tunnel (GRE/L2TPv3)
- **acces\_internet** qui donne accès au vrai internet aux postes
- **blocage\_vlan** qui bloque la communication entre des machines de VLANs différentes grâce à de la Policy Routing.

## Références

[https://p-fb.net/fileadmin/Infrastructure/2020\\_2021/cours\\_infra.pdf](https://p-fb.net/fileadmin/Infrastructure/2020_2021/cours_infra.pdf)

[https://fr.wikipedia.org/wiki/Layer\\_2\\_Tunneling\\_Protocol\\_Version\\_3](https://fr.wikipedia.org/wiki/Layer_2_Tunneling_Protocol_Version_3)

[https://fr.wikipedia.org/wiki/Virtual\\_Extensible\\_LAN](https://fr.wikipedia.org/wiki/Virtual_Extensible_LAN)

[https://fr.wikipedia.org/wiki/Multiprotocol\\_Label\\_Switching](https://fr.wikipedia.org/wiki/Multiprotocol_Label_Switching)

<https://www.netify.co.uk/learning/mpls-vs-vpn-ipsec>

<https://community.fs.com/blog/vpn-vs-mpls-difference.html>

<http://www.allgoodbits.org/articles/view/24>

<https://blog.scottlowe.org/2013/05/29/a-quick-introduction-to-linux-policy-routing/>