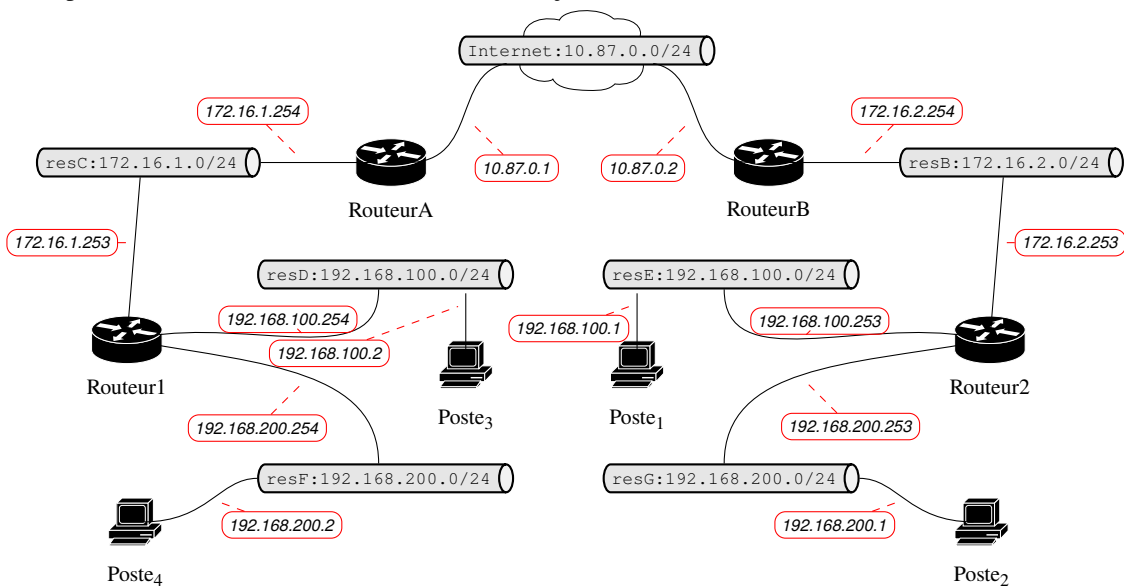


*Tunnel L2TPv3 sécurisé par IPsec pour la mise en œuvre de «trunking» de VLANs
Comparaison avec les VXLAN*

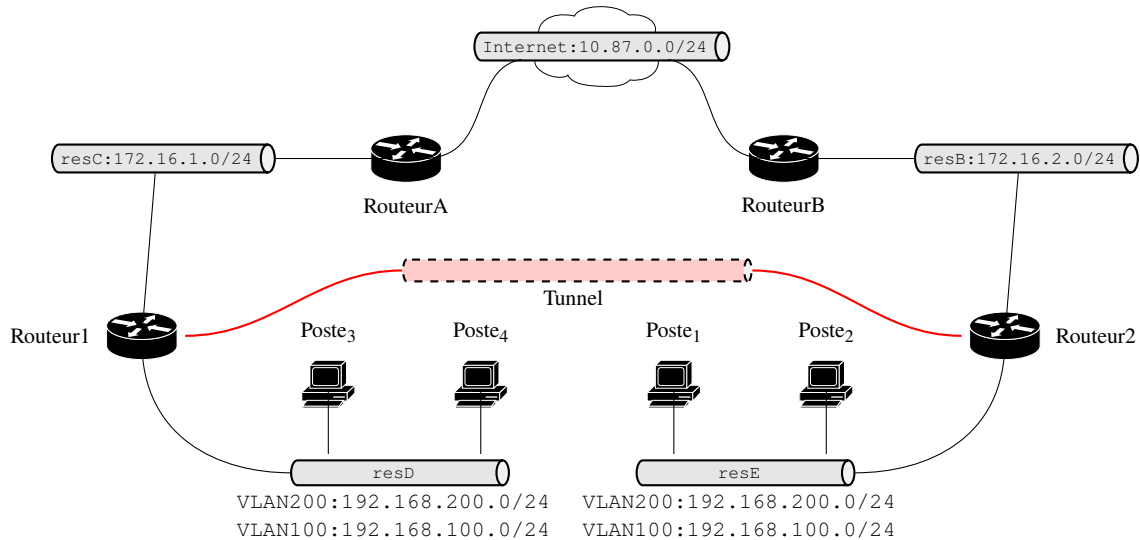
■ ■ ■ Présentation du projet

Le projet a pour but d'étudier le protocole L2TPv3, RFC 3931, pour faire des tunnels de niveau 2. On s'en servira pour faire du «trunking» de VLANs et de la mutualisation de service comme le DHCP. Enfin, à l'aide d'une configuration «intelligente», 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».

On dispose du réseau suivant où les réseaux sont disjoints au niveau 2 :



On voudrait arriver au schéma suivant où les réseaux sont liés au niveau 2 :



Un serveur DHCP sera mis en œuvre sur «Routeur1» afin de configurer les différents hôtes des deux VLANs (qu'ils soient ou non séparés par Internet).

Chaque poste et routeur sera mis en œuvre à l'aide d'un «netns».



Linux & VLANs

Dans ce projet, nous allons utiliser des VLANs afin de :

- créer deux réseaux locaux **différents** regroupant des machines connectées au travers des mêmes éléments actifs du réseaux (bridge/switch) avec des adresses de niveau 3 différentes (adresse IP) ;
- isoler les trafics de niveau 2 de ces réseaux pour qu'ils n'interfèrent pas entre eux : ARP, DHCP, « *Neighbor Discovery Protocol* », etc.
- multiplexer le trafic des deux réseaux au sein d'un même switch/bridge à la manière du « *VLAN trunking* » :
 - ◇ la trame est « étiquetée », « *VLAN Tagging* », pour indiquer à quel VLAN elle appartient (suivant le format 802.1Q) ;
 - ◇ la mise en place de l'étiquette ou sa suppression est réalisée soit par :
 - * l'élément actif du réseau (bridge/switch) et on parle de **VLAN de port** (sur un switch matériel le réglage est « *untagged* ») : la connexion ne permet d'atteindre qu'un seul VLAN et les trames ne contiennent pas d'étiquette à leur arrivée sur le port ;
 - * la machine connectée au réseau et on parle de réglage « *tagged* » : cette configuration est utile, par exemple, lorsque l'on connecte sur le même port un ordinateur et un téléphone IP, utilisant le protocole « *Voice Over IP* », dont le trafic n'ira pas dans le même VLAN et sera prioritisé suivant de la QoS.

Nous choisissons de faire du **VLAN de port**, qui offre la **sécurité maximale** tout en gardant la flexibilité du VLAN :

- ▷ la connexion n'appartient qu'à un seul VLAN ;
- ▷ les trames sont étiquetées dans le bridge/switch pour permettre le « *trunking* » ;
- ▷ les trames VLANs peuvent être multiplexées sur un lien pouvant être un **tunnel de niveau 2** ;
- ▷ les VLANs peuvent être répartis entre des sites, en employant le réseau IP pour assurer cette interconnexion.

Mise en place de VLAN de port sous Linux

- Ce que l'on veut faire :
 - ◇ ajouter l'étiquette VLAN sur les trames en sortie du netns ;
 - ◇ intégrer ce trafic dans un switch pour :
 - * les échanges avec les autres netns appartenant au même VLAN ;
 - * le passage dans le tunnel de niveau 2 pour relayer le trafic au travers du réseau IP vers les netns.
- Comment fonctionnent les VLANs sous Linux :
 - ◇ il est possible de gérer les VLANs comme sur un vrai switch avec Open vSwitch :
 - * VLAN de port : un port est associé à un seul VLAN, il peut alors :
 - ▷ laisser passer des trames non étiquetées dans le VLAN associé ;
 - ▷ filtrer ou non suivant la configuration du port des trames étiquetées ;
 - * réaliser de l'étiquetage VLAN au travers du « *trunking* » ;
 - * le VLAN s'exprime par l'ajout d'étiquette : « *VLAN tagged* » sur les trames ;
 - * le VLAN crée une nouvelle interface qui s'accroche à une interface existante :
 - ▷ lorsqu'une trame est envoyée sans étiquette depuis la nouvelle interface VLAN créée, elle apparaît avec une étiquette VLAN sur l'interface existante ;
 - ▷ lorsqu'une trame est reçue avec une étiquette VLAN sur l'interface existante, elle apparaît sur la nouvelle interface créée sans l'étiquette.
 - * il est possible d'accrocher plusieurs interfaces VLANs sur la même interface, **seulement** si elles appartiennent à des VLANs différents.
- Comment nous allons procéder :
 1. accrocher un VLAN sur l'interface du netns pour que le trafic en sortie du netns soit étiqueté (fonctionnement en VLAN de port) ;
 2. ajouter l'autre extrémité dans un switch (c'est par ce switch que le trafic sera retransmis par le tunnel).
 3. configurer le routage pour utiliser l'interface avec VLAN ;
- En conclusion :
 - ▷ tous les netns pourront communiquer entre eux s'ils sont connectés au même switch et appartiennent au même VLAN ;
 - ▷ le trafic étiqueté peut être « tunnelisé » au travers d'un tunnel de niveau 2.

■■■ Configuration des VLANs sous Linux

Pour ajouter le VLAN n°100 sur l'interface `rout1-eth1` :

- ▷ on choisira un nom pour l'interface, comme par exemple `rout1-eth1.100` pour rester compatible avec les notations utilisées par la commande `vconfig` qui était utilisée auparavant ;
- ▷ on utilisera la commande « `ip` » :

```
xterm
pef@cerberus:~$ sudo ip link
2: rout1-eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN
mode DEFAULT qlen 1000
    link/ether 00:0c:29:0f:31:a1 brd ff:ff:ff:ff:ff:ff
pef@cerberus:~$ sudo ip link add link rout1-eth1 name rout1-eth1.100 type vlan id 100
pef@cerberus:~$ ip link
2: rout1-eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN
mode DEFAULT qlen 1000
    link/ether 00:0c:29:0f:31:a1 brd ff:ff:ff:ff:ff:ff
4: rout1-eth1.100@eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
    link/ether 00:0c:29:0f:31:a1 brd ff:ff:ff:ff:ff:ff
pef@cerberus:~$ ip addr add 192.168.100.254/24 dev rout1-eth1.100
```

Les trames qui passeront par cette interface posséderont un « tag » de VLAN avec l'id 100.

*On note que l'interface initiale, ici `rout1-eth1`, ne dispose **pas de configuration IP**, mais c'est l'« interface » VLAN, ici `rout1-eth1.100` qui va être configurée en IP.*

- ▷ on peut ensuite donner une adresse IP à cette nouvelle interface :

```
xterm
pef@cerberus:~$ sudo ip addr add 192.168.1.1/24 brd 192.168.1.255 dev eth0.100
pef@cerberus:~$ sudo ip link set dev eth0.100 up
pef@cerberus:~$ ip address
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen
1000
    link/ether 00:0c:29:0f:31:a1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.127.213/24 brd 192.168.127.255 scope global eth0
    inet6 fe80::20c:29ff:fe0f:31a1/64 scope link
        valid_lft forever preferred_lft forever
4: eth0.100@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether 00:0c:29:0f:31:a1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.1/24 brd 192.168.1.255 scope global eth0.100
    inet6 fe80::20c:29ff:fe0f:31a1/64 scope link
        valid_lft forever preferred_lft forever
```

■ ■ ■ Retransmission de trames sous Linux

Pour la mise en place d'un tunnel niveau 2, c-à-d capable d'acheminer des trames Ethernet, il est nécessaire de pouvoir « récupérer » les trames circulant dans un réseau au travers d'une interface.

Les éléments nécessaires sont les suivants :

- il faut disposer d'une interface IF_1 **connectée dans le réseau** depuis lequel on veut récupérer les trames ;
- cette interface peut être au choix :
 - ◇ l'extrémité d'un lien de type « veth », l'autre extrémité étant connectée dans le réseau ;
 - ◇ une interface physique connectée à un réseau physique ;
- cette interface IF_1 doit être **active** mais **non configurée** pour IP (**pas d'adresse IP**) ;
- un **tunnel** est manipulable comme une interface : cette interface IF_2 doit être également **non configurée** ;
- l'interface du tunnel IF_2 doit être **reliée** à la première interface IF_1 :
 - ◇ on va utiliser un « bridge » dans lequel on va mettre IF_1 et IF_2 ;
 - ◇ toute trame que va « voir » IF_1 va également être vue par IF_2 , à condition que ces deux interfaces soient « up » et ne soient pas configurées pour IP (pas d'adresse IP configurée) (elles se comportent alors comme des interfaces en mode *promiscuous*, c-à-d qui relaie des trames qui ne sont pas à leur destination) ;
- pour permettre à la pile TCP/IP d'accéder au réseau connecté par IF_1 , il faut **configurer en IP** l'interface représentant le **bridge**.

Exemple pour la configuration de R1 sur le réseau de démo netlab des TPs du premier semestre :

```
❑ — xterm —
$ git clone https://git.unilim.fr/pierre-francois.bonnefoi/netlab.git

❑ — xterm —
❶ pef@gemu-ubuntu:~/netlab$ [r1] sudo ip l2tp add tunnel remote 172.16.2.253 local
172.16.1.253 encap ip tunnel_id 3000 peer_tunnel_id 4000
pef@gemu-ubuntu:~/netlab$ [r1] sudo ip l2tp add session tunnel_id 3000 session_id 1000
peer_session_id 2000
❷ pef@gemu-ubuntu:~/netlab$ [r1] sudo ip link set l2tpeth0 up
❸ pef@gemu-ubuntu:~/netlab$ [r1] sudo ip a flush dev r1-eth0
❹ pef@gemu-ubuntu:~/netlab$ [r1] sudo brctl addbr pont
❺ pef@gemu-ubuntu:~/netlab$ [r1] sudo brctl addif pont l2tpeth0
pef@gemu-ubuntu:~/netlab$ [r1] sudo brctl addif pont r1-eth0
❻ pef@gemu-ubuntu:~/netlab$ [r1] sudo ip link set pont up
```

- ❶ ⇒ Création du tunnel : l'interface l2tpeth0 est créée ;
- ❷ ⇒ activation de l'interface du tunnel ;
- ❸ ⇒ enlever la configuration ip de l'interface « r1-eth0 » ;
- ❹ ⇒ ajout d'un bridge nommé « pont » (il peut être nécessaire d'installer le paquet « bridge-utils ») ;
- ❺ ⇒ ajout des interfaces dans le bridge : celle du tunnel IF_2 et celle connectée au réseau IF_1 ;
- ❻ ⇒ activation de l'interface correspondant au bridge ;

Si on fait un ping vers une machine qui n'existe pas depuis la machine h1 :

```
❑ — xterm —
pef@gemu-ubuntu:~/netlab$ [h1] ping 192.168.10.5
```

On observe sur R1 :

```
❑ — xterm —
pef@gemu-ubuntu:~/netlab$ [r1] sudo tcpdump -i pont -lnvv
tcpdump: listening on tunnel, link-type EN10MB (Ethernet), snapshot length 262144
bytes
12:14:32.125292 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.10.5
tell 192.168.10.1, length 28
12:14:33.143768 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.10.5
tell 192.168.10.1, length 28
```

On observe également à la réception des paquets du tunnel sur R2 :

```
❑ — xterm —
pef@gemu-ubuntu:~/netlab$ [r2] sudo tcpdump -i r2-eth1 -lnvvX
tcpdump: listening on r2-eth1, link-type EN10MB (Ethernet), snapshot length 262144
bytes
12:51:28.605707 IP (tos 0x0, ttl 64, id 40674, offset 0, flags [none], proto
unknown (115), length 70)
  10.0.0.1 > 10.0.0.2: ip-proto-115 50
    0x0000: 4500 0046 9ee2 0000 4073 c760 0a00 0001  E..F....@s.`....
    0x0010: 0a00 0002 0000 07d0 0000 0000 ffff ffff  .....
    0x0020: ffff 1e1d 99b4 8a5d 0806 0001 0800 0604  .....].....
    0x0030: 0001 1e1d 99b4 8a5d c0a8 0a01 0000 0000  .....].....
    0x0040: 0000 c0a8 0a05  ....
```

■ ■ ■ Mise en place de tunnel L2TPv3

Pour configurer le tunnel L2TPv3, on a le choix entre une encapsulation dans IP ou bien dans UDP :

```
xterm
pef@cerberus:~$ sudo ip l2tp add tunnel remote 192.168.127.161 local 192.168.127.213
  encap ip tunnel_id 3000 peer_tunnel_id 4000
pef@cerberus:~$ sudo ip l2tp show tunnel
Tunnel 3000, encap IP
  From 192.168.127.213 to 192.168.127.161
  Peer tunnel 4000
```

Il faut configurer l'autre poste de la même façon mais en symétrique (hellhound ↔ cerberus) :

```
xterm
pef@hellhound:~$ sudo ip l2tp add tunnel local 192.168.127.161 remote 192.168.127.213
  encap ip tunnel_id 4000 peer_tunnel_id 3000
pef@hellhound:~$ sudo ip l2tp show tunnel
Tunnel 4000, encap IP
  From 192.168.127.161 to 192.168.127.213
  Peer tunnel 3000
```

On procède de la même façon pour établir une « session » :

```
xterm
pef@cerberus:~$ sudo ip l2tp add session tunnel_id 3000 session_id 1000 peer_session_id 2000
pef@cerberus:~$ ip link
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN mode DEFAULT qlen 1000
  link/ether 00:0c:29:0f:31:a1 brd ff:ff:ff:ff:ff:ff
6: l2tpeth0: <BROADCAST,MULTICAST> mtu 1492 qdisc noop state DOWN mode DEFAULT qlen 1000
  link/ether be:f0:1e:1e:e1:26 brd ff:ff:ff:ff:ff:ff
```

Et la version symétrique sur l'autre poste :

```
xterm
pef@hellhound:~$ sudo ip l2tp add session tunnel_id 4000 session_id 2000 peer_session_id 1000
pef@hellhound:~$ ip link
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN mode DEFAULT qlen 1000
  link/ether 00:0c:29:7f:2b:b6 brd ff:ff:ff:ff:ff:ff
4: l2tpeth0: <BROADCAST,MULTICAST> mtu 1492 qdisc noop state DOWN mode DEFAULT qlen 1000
  link/ether 4e:92:95:b4:81:74 brd ff:ff:ff:ff:ff:ff
```

On peut modifier, si nécessaire, la MTU de l'interface :

```
xterm
pef@cerberus:~$ sudo ip link set dev l2tpeth0 mtu 1500
pef@cerberus:~$ ip link
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN mode DEFAULT qlen 1000
  link/ether 00:0c:29:0f:31:a1 brd ff:ff:ff:ff:ff:ff
6: l2tpeth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
  link/ether be:f0:1e:1e:e1:26 brd ff:ff:ff:ff:ff:ff
```

Et sur l'autre poste :

```
xterm
pef@hellhound:~$ sudo ip link set dev l2tpeth0 mtu 1500
pef@hellhound:~$ ip link
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN mode DEFAULT qlen 1000
  link/ether 00:0c:29:7f:2b:b6 brd ff:ff:ff:ff:ff:ff
4: l2tpeth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
  link/ether 4e:92:95:b4:81:74 brd ff:ff:ff:ff:ff:ff
```

Travail à réaliser

Vous rédigerez un rapport au format PDF contenant des explications, des commandes, des configurations, des captures d'écran et des analyses de trames.

- 1 – Vous mettrez en œuvre le réseau conformément au schéma proposé en vous inspirant des TP déjà réalisés. Vous testerez le bon fonctionnement de l'encapsulation VLAN en « sniffant » le réseau à l'aide de « tcpdump ». Vous incorporerez dans votre rapport une capture **commentée** de trafic VLAN entre un hôte et un routeur (qui sont les interlocuteurs, où le trafic est-il sniffé, à qui appartient les adresses MAC, quelle est la nature du trafic, son sens d'échange et le protocole observé).
- 2 – Vous rédigerez une description/présentation rapide du protocole L2TPv3 et de la technologie des VXLANs, « *Virtual eXtensible Local Area Network* », RFC 7348. Vous complétez cette présentation :
 - a. en comparant les deux solutions ;
 - b. en étudiant la mise en œuvre dans Linux des VXLANs dans Open vSwitch ;
 - c. en étudiant les solutions de chiffrement du trafic L2TPv3 ou VXLAN ;
 - d. en comparant avec MPLS ces deux technologies.
- 3 – Vous établirez le tunnel L2TPv3 en mode encapsulation IP entre Routeur₁ et Routeur₂. Vous vérifierez que :
 - a. le protocole ARP fonctionne normalement pour la découverte des machines quelle que soit leur côté de connexion par rapport à Internet ;
 - b. le service DHCP que vous n'exécutez que sur « Routeur1 » peut être utilisé par Poste₁ ;
 - c. une connexion TCP à l'aide de socat entre Routeur1 et Poste₁ est possible à travers le tunnel.Pour chaque vérification vous ajouterez dans votre rapport des captures de trafic et de configuration d'interfaces permettant de justifier du bon fonctionnement de ces observations.

4 – Vous comparerez :

- a. la mise en œuvre de L2TPv3 :
 - ◇ en mode encapsulation IP ;
 - ◇ en mode encapsulation UDP.

Vous ferez une capture d'un paquet L2TPv3 dans chaque mode pour le protocole que vous vous voudrez (tcp, arp, icmp, etc.) et vous le détaillerez jusqu'à identifier précisément son contenu pour chacun de ses modes.

- b. Vous comparerez à un tunnel réalisé avec GRE en mode GRE-TAP (tunnel de niveau 2) :

```
xterm
$ sudo ip link add eoip1 type gretap remote 192.168.127.161 local 192.168.127.213 nopmtudisc
$ sudo ip link set dev eoip1 up
```

Pour passez d'un tunnel à l'autre, il suffit de mettre l'interface du tunnel à désactivé à « down » et l'autre à activer à « up ».

Quelles différences ? La MTU est-elle la même ? etc.

- 5 – En vous servant des fiches de TP, vous mettrez en place un chiffrement IPsec sur votre tunnel L2TPv3 en encapsulation IP. À l'aide de la commande « iperf » vous comparerez la vitesse de transfert entre « Routeur1 » et Poste₁ avec et sans chiffrement. Vous ajouterez dans votre rapport votre configuration, une capture commentée de trafic l2tpv3 avec protection IPsec et les rapports fournis par « iperf ».
- 6 – Accès Internet « intelligent » :
 - a. vous ajouterez un accès vers Internet en configurant le switch « Internet » et votre machine réelle ;
 - b. vous ajouterez l'accès à Internet sur Routeur1 pour les postes des deux VLANs, à l'aide de règles « iptables » ;
 - c. vous vérifierez que le trafic de Poste₁ passe bien par Routeur1 ;
 - d. vous regarderez comment en utilisant « dnsmasq », vous pouvez rediriger les postes Poste₁ et Poste₂, lors de leur configuration par DHCP, vers Routeur 2 au lieu de Routeur 1 pour optimiser l'accès à Internet.Vous ajouterez dans votre rapport la configuration de « dnsmasq » ainsi que des captures montrant que le trafic de Poste₁ vers Internet, passe bien par Routeur2 au lieu de Routeur1.

7 – Vous interdirez le trafic entre VLAN 100 et VLAN 200 :

- a. à l'aide de règles « iptables » ;
- b. à l'aide de la « Policy Routing ».

Vous ajouterez dans votre rapport les configurations utilisées.

Remise du travail

Le travail devra être déposé sur **Communities** sous forme d'une archive.

Cette archive contiendra :

- ★ tous les fichiers de configuration utilisés qui permettent de déployer le réseau ;
- ★ le rapport au format PDF contenant les captures d'écran demandées.

■■■ Remarque

Pour suivre l'exécution de vos commandes dans votre fichier « build_architecture » :

```
#!/bin/bash -x
# graphe : INFRA_PROJET LAB
# créer les différents namespaces
ip netns add postel
...
```

Ce qui donne :

```
xterm
pef@cube:~/INFRA_PROJET$ sudo ./build_architecture
+ ip netns add postel
...
```

L'option « -x » indique à bash de faire un affichage de chaque commande du script exécutée.

Vous pouvez alors faire des « cycles » entre la commande « clean » et la commande « build_architecture » pour vérifier la configuration de votre lab.