

FACULTÉ DES SCIENCES ET TECHNIQUES
DE LIMOGES



MASTER 1 INFORMATIQUE

Projet Infrastructure Réseaux

Réalisé par:

Amadou Oury DIALLO

Kilani WAJDI

1 Mise en oeuvre du Réseau et VLAN

Pour la mise en oeuvre du réseau nous nous sommes inspiré des TPs, et dans notre cas nous avons choisi de faire un routage automatique de type RIP sur les routeurs A et B et un routage manuel sur les routeurs 1 et 2 du réseau. Ci-dessous l'image générée par le script python de notre fichier build_architecture contenant les configurations des switches virtuels simulant les réseaux et des netns simulant les machines (routeur, poste)

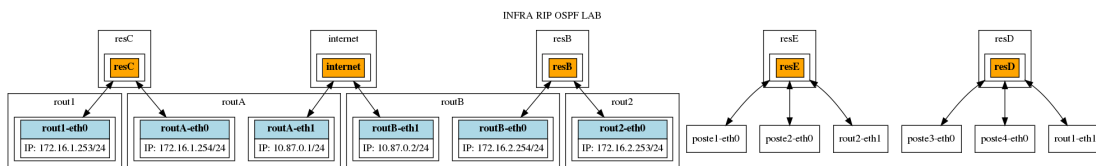


Figure 1: Architecture réseau

Nous constatons que les interfaces des rout1-eth1 et rout2-eth1 ne possède pas

1.1 Capture VLAN (Question 1)

Pour la capture nous avons fait le choix d'utiliser **tcpdump** avec l'option **-e vlan** :

```
root@amadink:~/PEF/INFRA/Projet# [rout1] tcpdump -lnvv -i rout1-eth1 -e vlan
tcpdump: listening on rout1-eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
01:00:03.305759 1a:bf:0a:cc:e3:9b > ca:d0:05:af:ba:5f, ethertype 802.1Q (0x8100), length 102: vlan 200,
p 0, ethertype IPv4, (tos 0x0, ttl 64, id 44059, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.200.76 > 192.168.100.38: ICMP echo request, id 23486, seq 1, length 64
01:00:03.305782 ca:d0:05:af:ba:5f > 82:c6:0f:52:01:44, ethertype 802.1Q (0x8100), length 102: vlan 100,
p 0, ethertype IPv4, (tos 0x0, ttl 63, id 44059, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.200.76 > 192.168.100.38: ICMP echo request, id 23486, seq 1, length 64
01:00:03.305952 82:c6:0f:52:01:44 > ca:d0:05:af:ba:5f, ethertype 802.1Q (0x8100), length 102: vlan 100,
p 0, ethertype IPv4, (tos 0x0, ttl 64, id 19838, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.100.38 > 192.168.200.76: ICMP echo reply, id 23486, seq 1, length 64
01:00:03.305959 ca:d0:05:af:ba:5f > 1a:bf:0a:cc:e3:9b, ethertype 802.1Q (0x8100), length 102: vlan 200,
p 0, ethertype IPv4, (tos 0x0, ttl 63, id 19838, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.100.38 > 192.168.200.76: ICMP echo reply, id 23486, seq 1, length 64
```

Figure 2: Trafic Vlan observé

Le trafic est capturé sur l'interface rout1-eth1(elle ne possédant pas de config IP) du routeur 1 et il s'agit d'un ping entre poste3 et poste 4. Nous

observons l'entête ethertype des paquets est **802.1Q (0x8100)** suivi de l'id du vlan (200 ou 100).

Nous observons aussi que la résolution ARP du ping s'effectue entre l'interface de poste4 (@mac : **1a:bf:0a:cc:e3:9b**) et l'interface du routeur rout1-eth1 (@mac : **ca:d0:05:af:ba:5f**) sur le premier paquets, puis sur le deuxième paquet on constate que l'interface du routeur rout1-eth1 fait à son tour une résolution ARP pour joindre le poste3 @mac : **82:c6:0f:52:01:44** et transmettre le paquet ICMP. En bref nous pouvons résumer que pour qu'une machine du vlan puisse joindre une autre elle utilise l' @mac de l'interface a laquelle elles sont attachés comme adresse mac source en réalisant une résolution ARP vers cette interface qui fera à son tour une résolution ARP vers la machine à joindre.

La figure ci-dessous montre les adresses mac des différents équipements

```

24: rout1-eth1@if23: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default qlen 1000
    link/ether ca:d0:05:af:ba:5f brd ff:ff:ff:ff:ff:ff link-netnsid 0
root@amadimk:~/PEF/INFRA/Projet# [rout1]
root@amadimk:~/PEF/INFRA/Projet 70x26
18: poste3-eth0@if17: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default qlen 1000
    link/ether 82:c6:0f:52:01:44 brd ff:ff:ff:ff:ff:ff link-netnsid 0
root@amadimk:~/PEF/INFRA/Projet# [poste3]
root@amadimk:~/PEF/INFRA/Projet
link/ether 1a:bf:0a:cc:e3:9b brd ff:ff:ff:ff:ff:ff link-netnsid 0
root@amadimk:~/PEF/INFRA/Projet# [poste4]

```

Figure 3: Adresse mac des équipements

2 Présentation de L2TPv3 & VxLANs (Question 2)

2.1 Présentation de L2TPv3

L2TPv3 (Layer 2 Tunneling Protocol Version 3) est un protocole d'encapsulation point à point d'un protocole quelconque de niveau 2 dans IP. Cela signifie qu'on peut tunneler les protocoles de couche 2 tels qu'Ethernet, Frame Relay, ATM, HDLC, PPP, etc. sur un réseau IP. Par exemple, supposons que nous ayons deux sites distants et une application nécessitant que les hôtes se trouvent sur le même sous-réseau. Avec L2TPv3, ce n'est pas un problème de relier deux sites distants en les plaçant dans le même domaine de diffusion / sous-réseau. L2TPv3 est une norme IETF (RFC3931) qui possède un numéro de protocole distinct (115) et combine certaines technologies :

- Cisco L2F (Layer 2 Forwarding)

- Microsoft Point to Point Tunneling Protocol (PPTP)

L2TPv3 peut être utilisé en deux modes d'encapsulation :

- mode en encapsulation IP
- mode en encapsulation UDP

2.2 Présentation de VxLAN

VxLAN est l'acronyme de Virtual eXtensible LAN standardisé à l'IETF en 2011 est un format d'encapsulation porté par Cisco et VMware ayant des fonctionnalités semblables aux VLANs mais avec quelques améliorations. VXLAN est un protocole de tunnelisation, qui permet d'étendre un réseau de couche 2 au dessus de réseaux routés. On identifie un réseau VXLAN par sa VNI (VXLAN Network Identifier). Celle-ci est codée sur 24 bits, ce qui donne 16777216 possibilités, on est alors bien loin de la limitation de 4096 induite par les VLANs.

Le fonctionnement de VXLAN est simple, il s'agit d'encapsuler une trame ethernet (couche 2 du modèle OSI), dans un datagramme UDP (couche 4). Les éléments chargés de cette encapsulation (et de la désencapsulation) sont appelés VTEP, pour VXLAN Tunnel End Point. Il suffit pour monter un réseau d'overlay VXLAN, que deux VTEP soient en mesure de communiquer en IP et en UDP, sur un numéro de port dédié (le port désigné par l'IANA est 4789, mais selon les implémentations on peut retrouver 8472 comme port par défaut, il est également possible d'en changer).

2.3 L2TPv3 vs VxLAN

Sachant que les 2 permettent d'encapsuler le niveau 2, nous remarquons tout de suite que la différence majeure entre les 2 réside dans la technologie VxLAN n'encapsule qu'en UDP tandis que la technologie L2TPv3 encapsule UDP et en IP de plus cette dernière (L2TPv3) permet d'étendre un même réseau du point de vue niveau se trouvant dans des emplacements distincts tandis que la première (VxLAN) permet de relier aussi des réseaux différents du point de vue niveau se trouvant dans des emplacements distincts.

2.4 VxLANs dans Open vSwitch

Pour configurer VxLAN avec Open vSwitch il suffit de procéder de la manière suivante :

```
$ ovs-vsctl add-br br0
$ ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=
    vxlan options:remote_ip=172.16.2.253 options:key=flow options
    :dst_port=8472
```

Explication :

1 - ajout d'un bridge.

2 - ajout du tunnel vxlan1 sur le bridge (l'autre machine remote doit être configurer de la même manière sauf pour l'adresse ip).

2.5 Chiffrement du trafic L2TPv3 ou VXLAN

Pour sécuriser les tunnels (L2TPv3 & VxLAN) nous pouvons utiliser un protocole de sécurité de niveau 3 au bout de chaque tunnel pour permettre de chiffrer de chaque bout de et de le déchiffré. Un des protocoles de ce types que nous pouvons utiliser est **IPsec (Internet Protocol Security, RFC 2401)** qui est un protocole de la couche 3 du modèle OSI, tout comme IP; permettant de chiffrées (confidentialité) et protégées (intégrité) les données transitant dans 2 extrémités sont authentifiées.

2.6 L2TPv3 & VXLAN vs MPLS

3 Tunnel L2TPv3

Pour la mise en place du tunnel vous le trouverez le config dans les fichiers : **build_tunnel_ip_vlan** en mode encapsulation IP et **build_tunnel_udp_vlan** en mode encapsulation UDP.

Pour construire le réseaux il suffit d'exécuter le script **build_architecture** suivi ensuite d'un des fichiers ci-dessus selon le type d'encapsulation que nous voulons.

3.1 Vérification ARP en mode IP

Pour la vérification d'arp nous avons lancer notre tcpdump sur l'interface **tunnel** du routeur et nous avons lancer un ping entre les hôtes **poste1**

et **poste3** se trouvant chacun d'un coté d'internet et la capture ci-dessous montre que le protocole fonctionne entre les deux machines.

```

root@amadink:~/PEF/INFRA/Projet142x7
tcpdump: listening on tunnel, link-type EN10MB (Ethernet), capture size 262144 bytes
23:27:51.050904 c6:d9:9a:a7:4b:b2 > ff:ff:ff:ff:ff:ff, ethertype 802.1Q (0x8100), length 46: vlan 100, p 0, ethertype ARP, Ethernet (len 6), I
Pv4 (len 4), Request who-has 192.168.100.58 tell 192.168.100.38, length 28
23:27:51.052579 fa:e0:77:07:a7:55 > c6:d9:9a:a7:4b:b2, ethertype 802.1Q (0x8100), length 46: vlan 100, p 0, ethertype ARP, Ethernet (len 6), I
Pv4 (len 4), Reply 192.168.100.58 is-at fa:e0:77:07:a7:55, length 28
23:27:51.053064 c6:d9:9a:a7:4b:b2 > fa:e0:77:07:a7:55, ethertype 802.1Q (0x8100), length 102: vlan 100, p 0, ethertype IPv4, (tos 0x0, ttl 64,
id 43065, offset 0, flags [none], proto ICMP, length 64)

root@amadink:~/PEF/INFRA/Projet70x30
root@amadink:~/PEF/INFRA/Projet# [poste3] ping 192.168.100.58
PING 192.168.100.58 (192.168.100.58) 56(84) bytes of data.
64 bytes from 192.168.100.58: icmp_seq=1 ttl=64 time=3.11 ms
64 bytes from 192.168.100.58: icmp_seq=2 ttl=64 time=0.232 ms
^C
--- 192.168.100.58 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.232/1.672/3.112/1.440 ms
root@amadink:~/PEF/INFRA/Projet# [poste3]

root@amadink:~/PEF/INFRA/Projet70x30
root@amadink:~/PEF/INFRA/Projet# [poste1] ip a show poste1-eth0.100
2: poste1-eth0.100@poste1-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu
1500 qdisc noqueue state UP group default qlen 1000
    link/ether fa:e0:77:07:a7:55 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::f8e0:77ff:fe07:a755/64 scope link
        valid_lft forever preferred_lft forever
    inet 192.168.100.58/24 brd 192.168.100.255 scope global poste1-eth
0.100
        valid_lft forever preferred_lft forever

```

Figure 4: Vérification arp

3.2 Vérification du service DHCP

Les configurations du serveur dhcp sont présent dans le fichier **build_tunnel_ip_vlan**. Nous lançons un tcpdump sur l'interface tunnel du routeur 1 et lançons **dhclient poste1-eth0.100** sur la machine poste 1 pour qu'elle demande l'obtention d'une adresse ip comme on peut le constater ci-dessous le dhcp marche bien.

```

root@amadink:~/PEF/INFRA/Projet70x22
root@amadink:~/PEF/INFRA/Projet# [route1] tcpdump -lnvv -i tunnel -e vl
an
tcpdump: listening on tunnel, link-type EN10MB (Ethernet), capture siz
e 262144 bytes
23:55:37.531613 fa:e0:77:07:a7:55 > ff:ff:ff:ff:ff:ff, ethertype 802.1
Q (0x8100), length 346: vlan 100, p 0, ethertype IPv4, (tos 0x10, ttl
128, id 0, offset 0, flags [none], proto UDP (17), length 328)
    0.0.0.0.68 > 255.255.255.255:67: [udp sum ok] BOOTP/DHCP, Request
from fa:e0:77:07:a7:55, length 300, xid 0x74023d7e, Flags [none] (0x00
00)

Client-Ethernet-Address fa:e0:77:07:a7:55
Vendor-rfc1048 Extensions
Magic Cookie 0x63825363
DHCP-Message Option 53, length 1: Discover
Requested-IP Option 50, length 4: 192.168.100.58
Hostname Option 12, length 7: "amadink"
Parameter-Request Option 55, length 13:
    Subnet-Mask, BR, Time-Zone, Default-Gateway
    Domain-Name, Domain-Name-Server, Option 119, Hostname
    Netbios-Name-Server, Netbios-Scope, MTU, Classless-Stat

c-Route
NTP

amadink@amadink:~/PEF/INFRA/Projet$ [poste1] ip a show poste1-eth0.100
2: poste1-eth0.100@poste1-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu
1500 qdisc noqueue state UP group default qlen 1000
    link/ether fa:e0:77:07:a7:55 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::f8e0:77ff:fe07:a755/64 scope link
        valid_lft forever preferred_lft forever
    inet 192.168.100.58/24 brd 192.168.100.255 scope global poste1-eth
0.100
        valid_lft forever preferred_lft forever
    inet6 fe80::f8e0:77ff:fe07:a755/64 scope link
        valid_lft forever preferred_lft forever
amadink@amadink:~/PEF/INFRA/Projet$ [poste1] sudo dhclient poste1-eth0
.100
amadink@amadink:~/PEF/INFRA/Projet$ [poste1] ip a show poste1-eth0.100
2: poste1-eth0.100@poste1-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu
1500 qdisc noqueue state UP group default qlen 1000
    link/ether fa:e0:77:07:a7:55 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.58/24 brd 192.168.100.255 scope global poste1-eth
0.100
        valid_lft forever preferred_lft forever
    inet6 fe80::f8e0:77ff:fe07:a755/64 scope link
        valid_lft forever preferred_lft forever
amadink@amadink:~/PEF/INFRA/Projet$ [poste1]

```

Figure 5: Dhcp prise en charge du poste 1

```

amadlnk@amadlnk:~/PEF/INFRA/Projets$ dnsmasq-dhcp: DHCPDISCOVER(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPOFFER(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPDISCOVER(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPOFFER(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPREQUEST(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPACK(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55 amadlnk
dnsmasq-dhcp: ne donne pas de nom amadlnk au bail DHCP de 192.168.100.58 parce-que le nom existe dans /etc/hosts avec l'adresse 127.0.1.1
dnsmasq-dhcp: DHCPDISCOVER(tunnel.100) 192.168.100.38 c6:d9:9a:a7:4b:b2
dnsmasq-dhcp: DHCPOFFER(tunnel.100) 192.168.100.38 c6:d9:9a:a7:4b:b2
dnsmasq-dhcp: DHCPDISCOVER(tunnel.100) 192.168.100.38 c6:d9:9a:a7:4b:b2
dnsmasq-dhcp: DHCPOFFER(tunnel.100) 192.168.100.38 c6:d9:9a:a7:4b:b2
dnsmasq-dhcp: DHCPREQUEST(tunnel.100) 192.168.100.38 c6:d9:9a:a7:4b:b2
dnsmasq-dhcp: DHCPACK(tunnel.100) 192.168.100.38 c6:d9:9a:a7:4b:b2 amadlnk
dnsmasq-dhcp: ne donne pas de nom amadlnk au bail DHCP de 192.168.100.38 parce-que le nom existe dans /etc/hosts avec l'adresse 127.0.1.1
dnsmasq-dhcp: DHCPREQUEST(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPACK(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55 amadlnk
dnsmasq-dhcp: ne donne pas de nom amadlnk au bail DHCP de 192.168.100.58 parce-que le nom existe dans /etc/hosts avec l'adresse 127.0.1.1
dnsmasq-dhcp: DHCPREQUEST(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPACK(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55 amadlnk
dnsmasq-dhcp: ne donne pas de nom amadlnk au bail DHCP de 192.168.100.58 parce-que le nom existe dans /etc/hosts avec l'adresse 127.0.1.1
dnsmasq-dhcp: DHCPRELEASE(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPDISCOVER(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPOFFER(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPDISCOVER(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPOFFER(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPREQUEST(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPACK(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55 amadlnk
dnsmasq-dhcp: ne donne pas de nom amadlnk au bail DHCP de 192.168.100.58 parce-que le nom existe dans /etc/hosts avec l'adresse 127.0.1.1
dnsmasq-dhcp: DHCPRELEASE(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPDISCOVER(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPOFFER(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPREQUEST(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55
dnsmasq-dhcp: DHCPACK(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55 amadlnk
dnsmasq-dhcp: ne donne pas de nom amadlnk au bail DHCP de 192.168.100.58 parce-que le nom existe dans /etc/hosts avec l'adresse 127.0.1.1
dnsmasq-dhcp: DHCPRELEASE(tunnel.100) 192.168.100.58 fa:e0:77:07:a7:55

```

Figure 6: Rêquete dhcp traité par dnsmasq qui simule le serveur dhcp

La capture ci-dessous montre que toutes les machines prennent un **default** l'interface vlan du routeur 1 car elles ont obtenu leur adresse ip par le serveur dhcp qui tourne sur ces interfaces dont **tunnel.100 : 192.168.100.254** pour le vlan 100 et **tunnel.200 : 192.168.200.254** pour le vlan 200.

<pre> root@amadimk: ~/PEF/INFRA/Projekt 70x5 root@amadimk:~/PEF/INFRA/Projekt# [post4] ip r default via 192.168.200.254 dev post4-eth0.200 192.168.200.0/24 dev post4-eth0.200 proto kernel scope link src 192.1 68.200.76 root@amadimk:~/PEF/INFRA/Projekt# [post4] </pre>	<pre> amadimk@amadimk: ~/PEF/INFRA/Projekt 70x5 amadimk@amadimk:~/PEF/INFRA/Projekt\$ [post1] ip r default via 192.168.100.254 dev post1-eth0.100 192.168.100.0/24 dev post1-eth0.100 proto kernel scope link src 192.1 68.100.58 amadimk@amadimk:~/PEF/INFRA/Projekt\$ [post1] </pre>
<pre> root@amadimk: ~/PEF/INFRA/Projekt 70x31 root@amadimk:~/PEF/INFRA/Projekt# [post3] ip r default via 192.168.100.254 dev post3-eth0.100 192.168.100.0/24 dev post3-eth0.100 proto kernel scope link src 192.1 68.100.38 root@amadimk:~/PEF/INFRA/Projekt# [post3] </pre>	<pre> root@amadimk:~/PEF/INFRA/Projekt 70x31 root@amadimk:~/PEF/INFRA/Projekt# [post2] ip r default via 192.168.200.254 dev post2-eth0.200 192.168.200.0/24 dev post2-eth0.200 proto kernel scope link src 192.1 68.200.54 root@amadimk:~/PEF/INFRA/Projekt# [post2] </pre>

Figure 7: default route des postes fournis par dhcp

3.3 Connexion TCP routeur 1 poste 1 via tunnel

Nous lançons un `tcpdump` sur l'interface du tunnel `l2peth0` et une `socat` serveur au niveau du routeur 1 puis une `socat` cliente au niveau du poste 1. Sur le `tcpdump` nous pouvons constater que les paquets empruntent bien le

tunnel et nous constatons aussi l'établissement de la connexion tcp avec les flags : **SYN - SYN/ACK** et **ACK** représenté respectivement sur les paquets par **[S]** - **[S.]** et **[.]**

The screenshot shows a terminal window with two panes. The top pane displays the output of a tcpdump command, showing several TCP packets between 192.168.100.58 and 192.168.100.254. The bottom pane shows the setup of a socat listener on port 6789, with messages from 'poste 1' and 'routeur 1' being received.

```

root@amadimk: ~/PEF/INFRA/Projet 142x18
root@amadimk:~/PEF/INFRA/Projet# [rout1] tcpdump -lnvv -i l2tpeth0
tcpdump: listening on l2tpeth0, link-type EN10MB (Ethernet), capture size 262144 bytes
00:46:43.611086 IP (tos 0x0, ttl 64, id 17863, offset 0, flags [DF], proto TCP (6), length 60)
  192.168.100.58.52390 > 192.168.100.254.6789: Flags [S], cksum 0xe78d (correct), seq 200758662, win 29200, options [mss 1460,sackOK,TS val 282020445 ecr 0,nop,wscale 10], length 0
00:46:43.611170 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 60)
  192.168.100.254.6789 > 192.168.100.58.52390: Flags [S.], cksum 0x995d (correct), seq 2170688602, ack 200758663, win 28120, options [mss 1418,sackOK,TS val 429761319 ecr 282020445,nop,wscale 10], length 0
00:46:43.612008 IP (tos 0x0, ttl 64, id 17864, offset 0, flags [DF], proto TCP (6), length 52)
  192.168.100.58.52390 > 192.168.100.254.6789: Flags [S.], cksum 0x35bd (correct), seq 1, ack 1, win 29, options [nop,nop,TS val 282020446 ecr 429761319], length 0
00:46:47.544149 IP (tos 0x0, ttl 64, id 17865, offset 0, flags [DF], proto TCP (6), length 60)
  192.168.100.58.52390 > 192.168.100.254.6789: Flags [P.], cksum 0x73f7 (correct), seq 1:9, ack 1, win 29, options [nop,nop,TS val 282024378 ecr 429761319], length 8
00:46:47.544202 IP (tos 0x0, ttl 64, id 1050, offset 0, flags [DF], proto TCP (6), length 52)
  192.168.100.254.6789 > 192.168.100.58.52390: Flags [S.], cksum 0x16fd (correct), seq 1, ack 9, win 28, options [nop,nop,TS val 429765252 ecr 282024378], length 0
00:46:48.418548 IP (tos 0x0, ttl 64, id 37868, offset 0, flags [DF], proto UDP (17), length 328)

root@amadimk:~/PEF/INFRA/Projet 70x19
root@amadimk:~/PEF/INFRA/Projet# [rout1] socat tcp-listen:6789 -
coucou je suis poste 1
bonjour poste1 je suis routeur 1

root@amadimk:~/PEF/INFRA/Projet 70x19
root@amadimk:~/PEF/INFRA/Projet# [poste1] socat - tcp:192.168.100.254:6789
coucou je suis poste 1
bonjour poste1 je suis routeur 1

```

Figure 8: socat TCP entre routeur 1 et poste 1

4 Tunnel L2TPv3 IP/UDP vs GRE

4.1 Tunnel L2TPv3 en mode IP vs UDP

Pour comprendre la différence entre une tunnelisation L2TPv3 en mode IP et UDP nous avons sniffé des paquets sur l'interface rout2-eth0 du routeur 1 et avons établi une connexion tcp entre routeur 1 et poste 1 dans chacun des modes, puis ensuite nous avons ouverts les captures du trafic avec wireshark pour analyser les 2 captures.

par ailleurs il est à préciser que lors des configurations du L2TPv3 en mode UDP **Tunnel L2Tv3 en mode IP**

No.	Time	Source	Destination	Protocol	Length	Info
7	4.096302	172.16.2.253	172.16.1.253	0x0000	116	D[S:0x3E8]
8	4.595425	172.16.2.253	172.16.1.253	0x0000	120	D[S:0x3E8]
9	4.595484	172.16.1.253	172.16.2.253	0x0000	112	D[S:0x7D0]
10	6.912369	172.16.2.253	172.16.1.253	0x0000	88	D[S:0x3F81]

▶ Frame 8: 120 bytes on wire (960 bits), 120 bytes captured (960 bits)
 ▶ Ethernet II, Src: c2:8d:7d:40:7d:9a (c2:8d:7d:40:7d:9a), Dst: de:0a:9e:d6:dd:61 (de:0a:9e:d6:dd:61)
 ▶ Internet Protocol Version 4, Src: 172.16.2.253, Dst: 172.16.1.253
 ▶ Layer 2 Tunneling Protocol version 3
 ▶ Cisco HDLC
 ▶ Data (78 bytes)

```

0000  de 0a 9e d6 dd 61 c2 8d 7d 40 7d 9a 08 00 45 00  ....a.. }@}...E-
0010  00 6a a9 81 09 00 3e 73 75 85 ac 10 02 fd ac 10  .j....>s u.....
0020  01 fd 00 00 03 e8 00 00 00 00 42 d5 36 07 1c e9  .....B.6....
0030  4a 7b c1 02 ed 5b 81 00 00 04 08 00 45 00 00 3c  J{...[...d..E-
0040  71 52 40 00 40 06 7f 0b c0 a8 64 0f c0 a8 64 fe  qR@.0...d..d..
0050  c5 40 1a 85 e7 aa ea 95 07 c4 f8 51 80 18 00 1d  .@.....:Q.....
0060  db 21 00 00 01 01 08 0a bd b0 95 c9 3e 95 5a 8a  !.....>Z-
0070  62 6f 6e 6a 6f 75 72 0a  ....bonjour
  
```

Figure 9: Encapsulation tunnel l2tpv3 mode IP

Nous remarquons le protocole IP qui encapsule un protocole L2TPv3 avec wireshark, tandis que qu'avec tcpdump ce protocole est marqué **"unknown"**

Tunnel L2Tv3 en mode IP

No.	Time	Source	Destination	Protocol	Length	Info
2	0.000064	172.16.1.253	172.16.2.253	SKYPE	132	Unknown_0
3	0.000548	172.16.2.253	172.16.1.253	SKYPE	124	Unknown_0
4	2.709235	172.16.2.253	172.16.1.253	SKYPE	130	Unknown_0
5	2.709298	172.16.1.253	172.16.2.253	SKYPE	124	Unknown_0

▶ Frame 4: 130 bytes on wire (1040 bits), 130 bytes captured (1040 bits)
 ▶ Ethernet II, Src: 7a:fe:48:6d:35:d4 (7a:fe:48:6d:35:d4), Dst: be:16:3c:45:62:e1 (be:16:3c:45:62:e1)
 ▶ Internet Protocol Version 4, Src: 172.16.2.253, Dst: 172.16.1.253
 ▶ User Datagram Protocol, Src Port: 2094, Dst Port: 2093
 ▶ SKYPE

```

0000  be 16 3c 45 62 e1 7a fe 48 6d 35 d4 08 00 45 00  ...<Eb.Z- Hm5...E-
0010  00 74 fb ff 00 00 3e 11 23 5f ac 10 02 fd ac 10  .t....> #_.....
0020  01 fd 08 2e 08 2d 00 60 00 00 00 03 00 00 00 00  .....X-
0030  03 e8 00 00 00 00 1e 99 ac f4 fc 83 e6 e3 58 b8  .....d..E...j@-
0040  04 5e 81 00 00 64 08 00 45 00 00 3a d4 6a 40 00  @.....dL...d..n-
0050  40 06 1b b8 c0 a8 64 4c c0 a8 64 fe af 6e 1a 85  fb@.S.....L...
0060  66 62 40 da 53 0f 19 bf 80 18 00 1d 4c 14 00 00  ....G...p-salu
0070  01 01 08 0a 87 47 ff ea b6 0e 70 c1 73 61 6c 75  t-
0080  74 0a
  
```

Figure 10: Encapsulation tunnel l2tpv3 mode UDP

En udp nous remarquons clairement que le protocole IP encapsule le protocole UDP nous voyons les ports on que nous avons indiqué lors de la configuration du tunnel en udp : **2094 et 2093**. Pour conclure nous pouvons donc dire que la différence entre les deux modes du tunnel L2TPv3 réside dans la manière dont il procède pour l'encapsulation notamment dans le cas du mode IP il encapsule le protocole L2TPv3 lui-même qui est encapsuler à son tour par le protocole IP pour faire la différence

ainsi entre les deux (mode IP et protocole IP) tandis qu'en mode UDP le protocole UDP est directement encapsuler par le protocole IP.
Par ailleurs il est à préciser que pour voir cette différence il faut sniffer les paquets n'ont pas dans le tunnel mais en dehors.

4.2 Tunnel L2TPv3 vs Tunnel GRE

GRE (Generic Routing Encapsulation) est un simple protocole d'encapsulation IP, il est utilisé pour envoyer un paquet d'un réseau a un autre sans devoir le traiter comme un paquet IP par les routeurs intervenants. le L2TP permet l'établissement et le maintien de l'émulation des PPP session qui est en une différence par rapport à un Tunnel GRE, aussi, les MTU sont différentes puisque le tunnel L2TPv3 rajoute **40 octets** en plus des 40 octets de l'entete TCP/IP (par exemple pour pouvoir supporter des trames de 1500 octets sans fragmentation des paquets il faudra configurer l'interface pour une MTU égale à 1500 - 40 (TCP/IP header) - 40 (L2Tv3 header) = 1420 octets) tandis que la MTU pour un tunnel GRE est généralement fixe à **1476 octets** par exemple (si un paquet de 1500 octets arrive sur un tunnel GRE, il est fragmenté en deux dont un premier paquet de 1476 + 24 octets (GRE header) = 1500 octets et un second de 44 + 24 octets (GRE header) = 68 octets avant d'être acheminer.) donc en conclusion l'entête GRE est 24 octets et celle de L2TPv3 octets.

No.	Time	Source	Destination	Protocol	Length	Info
5	7.292466	192.168.100.254	192.168.100.87	TCP	116	6787 → 39218 [S]
6	7.292717	192.168.100.87	192.168.100.254	TCP	108	39218 → 6787 [A]
7	10.784971	192.168.100.87	192.168.100.254	TCP	116	39218 → 6787 [P]
8	10.785037	192.168.100.254	192.168.100.87	TCP	108	6787 → 39218 [A]

Frame 5: 116 bytes on wire (928 bits), 116 bytes captured (928 bits)

- Ethernet II, Src: 56:54:fa:2f:c7:f9 (56:54:fa:2f:c7:f9), Dst: 2a:45:c9:11:c1:1d (2a:45:c9:11:c1:1d)
- Internet Protocol Version 4, Src: 172.16.1.253, Dst: 172.16.2.253
- Generic Routing Encapsulation (Transparent Ethernet bridging)
- Ethernet II, Src: 26:b2:d0:a1:f9:ca (26:b2:d0:a1:f9:ca), Dst: 86:54:56:41:b7:a6 (86:54:56:41:b7:a6)
- 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 100
- Internet Protocol Version 4, Src: 192.168.100.254, Dst: 192.168.100.87
- Transmission Control Protocol, Src Port: 6787, Dst Port: 39218, Seq: 0, Ack: 1, Len: 0

Figure 11: Capture d'un trafic TCP empruntant le tunnel GRE

5 Configuration IPsec

Pour configurer notre réseau avec IPsec afin de chiffrer le trafic empruntant le tunnel, nous avons procédé à une configuration inspiré des TP's ainsi, le

Le fichier `build_ipsec_rout1_rout2` contient cette configuration qui s'applique sur les interfaces d'entrées du tunnel de chaque routeur (1 et 2).

```

root@amadink: ~/PEF/INFRA/Projet# [rout1] tcpdump -lnvv -i rout1-eth0
tcpdump: listening on rout1-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
22:13:51.415625 IP (tos 0x0, ttl 62, id 16930, offset 0, flags [none], proto ESP (50), length 160)
  172.16.2.253 > 172.16.1.253: ESP(spi=0x12345678,seq=0x17), length 140
22:13:51.416064 IP (tos 0x0, ttl 64, id 17126, offset 0, flags [none], proto ESP (50), length 160)
  172.16.1.253 > 172.16.2.253: ESP(spi=0x12345678,seq=0x19), length 140
22:13:52.416948 IP (tos 0x0, ttl 62, id 16936, offset 0, flags [none], proto ESP (50), length 160)
  172.16.2.253 > 172.16.1.253: ESP(spi=0x12345678,seq=0x18), length 140
22:13:52.417122 IP (tos 0x0, ttl 64, id 17295, offset 0, flags [none], proto ESP (50), length 160)
  172.16.1.253 > 172.16.2.253: ESP(spi=0x12345678,seq=0x1a), length 140
22:13:53.448975 IP (tos 0x0, ttl 62, id 17027, offset 0, flags [none], proto ESP (50), length 160)
  172.16.2.253 > 172.16.1.253: ESP(spi=0x12345678,seq=0x19), length 140
root@amadink: ~/PEF/INFRA/Projet 70x23
root@amadink: ~/PEF/INFRA/Projet

```

Figure 12: Capture du trafic utilisant Ipsec pour chiffrement

Ci-dessus on peut remarquer que le trafic contient ESP (Encapsulation Security Protocol) et nous remarquons que chaque paquet est identifiée par un SPI (security parameters index) unique (32 bits) que nous avons choisi `spi = 0x12345678`. Par ailleurs nous rappelons que ce trafic concerne un ping entre poste 1 et poste 2 que nous n'arriverons pas à identifier ici par l'utilisation d'IPsec qui chiffre ce trafic.

```

root@amadink: ~/PEF/INFRA/Projet 70x26
root@amadink: ~/PEF/INFRA/Projet# [rout1] iperf -s
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
[ 4] local 192.168.100.254 port 5001 connected with 192.168.100.87 port 49820
[ ID] Interval      Transfer    Bandwidth
[ 4] 0.0-10.0 sec  570 MBytes  477 Mbits/sec
^Croot@amadink: ~/PEF/INFRA/Projet# [rout1]

root@amadink: ~/PEF/INFRA/Projet 70x26
root@amadink: ~/PEF/INFRA/Projet# [poste1] iperf -c 192.168.100.254
Client connecting to 192.168.100.254, TCP port 5001
TCP window size: 45.0 KByte (default)
[ 3] local 192.168.100.87 port 49820 connected with 192.168.100.254 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.0 sec  570 MBytes  478 Mbits/sec
root@amadink: ~/PEF/INFRA/Projet# [poste1]

```

Figure 13: Rapport de performance avec iperf sans utilisation du chiffrement IPsec

On peut remarquer que pour un intervalle de **10s** iperf a reçu à transférer entre poste 1 et routeur 1 **570 MBytes** de paquets TCP soit **470 MBytes/s** de Bande passante.

```

root@amadink: ~/PEF/INFRA/Projet 70x23
root@amadink:~/PEF/INFRA/Projet# [rout1] iperf -s
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
[ 4] local 192.168.100.254 port 5001 connected with 192.168.100.87 port 49832
[ ID] Interval      Transfer    Bandwidth
[ 4] 0.0-10.0 sec  208 MBytes  174 Mbits/sec

root@amadink:~/PEF/INFRA/Projet# [postel] iperf -c 192.168.100.254
Client connecting to 192.168.100.254, TCP port 5001
TCP window size: 45.0 KByte (default)
[ 3] local 192.168.100.87 port 49832 connected with 192.168.100.254 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.0 sec  208 MBytes  174 Mbits/sec
root@amadink:~/PEF/INFRA/Projet# [postel]

```

Figure 14: Rapport de performance avec iperf avec utilisation du chiffrement IPsec

Ici, remarquons que pour un intervalle de **10s** iperf à reçu à transférer entre poste 1 et routeur 1 **208 MBytes** de paquets TCP soit **174 MBytes/s** de Bande passante qui est plus que la moitié en utilisation sans IPsec.

6 Accès Internet « intelligent »

6.1 Accès Internet

Pour Permettre l'accès internet à notre réseau, nous avons activé au niveau de notre machine root l'interface du bridge internet et nous lui avons attribué une adresse d'un réseau appartenant à notre architecture **10.87.0.253** ensuite nous avons ajouté une règle firewall à notre machine root pour faire du SNAT de notre réseau 10.87.0.0/24 provenant de cette interface et nous avons aussi activé la capacité à notre machine de faire de forwarding qui est désactivé par défaut afin de lui permettre rediriger nos paquets provenant de notre architecture à la manière d'un routeur. A partir de là on peut donc remarquer que nous avons accès à internet à partir du routeurA et du routeurB.

```

root@amadink:~/PEF/INFRA/Projet 70x14
root@amadink:~/PEF/INFRA/Projet# [routA] ping -c 3 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=116 time=294 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=61.0 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=49.2 ms
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 49.275/134.968/294.564/112.954 ms
root@amadink:~/PEF/INFRA/Projet# [routA]

root@amadink:~/PEF/INFRA/Projet 70x14
root@amadink:~/PEF/INFRA/Projet# [routB] ping -c 3 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=116 time=137 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=155 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=61.9 ms
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 61.937/118.293/155.831/40.578 ms
root@amadink:~/PEF/INFRA/Projet# [routB]

```

Figure 15: Autorisation d'accès à internet

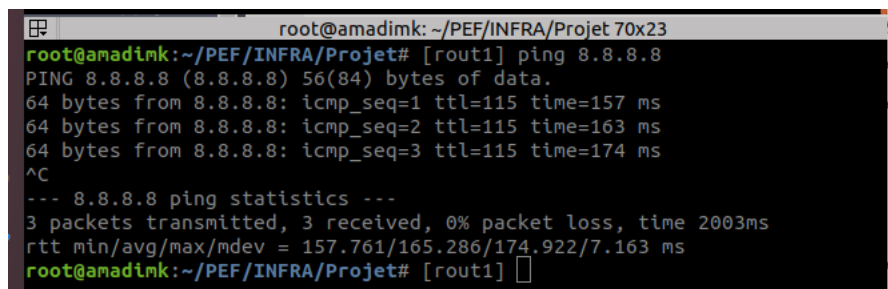
Toutes les règles et les configurations à effectuer se trouvent dans le fichier : **build_internet**

6.2 Accès Internet pour routeur 1 avec iptables

Pour ce faire il suffit de configurer le routeur A avec des règles firewall d'iptables en faisant du SNAT comme suit :

```
$ ip netns exec routA iptables -t nat -A POSTROUTING -s
172.16.1.0/24 -j MASQUERADE
$ ip netns exec routA iptables -t nat -A POSTROUTING -s
192.168.100.0/24 -j MASQUERADE
$ ip netns exec routA iptables -t nat -A POSTROUTING -s
192.168.200.0/24 -j MASQUERADE
```

Parallèlement pour autoriser internet sur le routeur 2 il suffit de faire même configuration sauf pour la première règle où il faut substituer le réseau **172.16.1.0/24** par **172.16.2.0/24**

A terminal window titled 'root@amadimk: ~/PEF/INFRA/Projet 70x23' shows a user running a ping command from a router interface. The output shows three successful ping packets to 8.8.8.8 with varying times and no packet loss. The user then presses Ctrl-C to stop the ping.

```
root@amadimk:~/PEF/INFRA/Projet 70x23
root@amadimk:~/PEF/INFRA/Projet# [rout1] ping 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=115 time=157 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=115 time=163 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=115 time=174 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 157.761/165.286/174.922/7.163 ms
root@amadimk:~/PEF/INFRA/Projet# [rout1]
```

Figure 16: Autorisation d'accès à internet pour routeur 1

6.3 Accès Internet pour poste 1

Sachant que poste 1 à obtenu sa configuration IP par DHCP donc sa route par défaut devrait etre l'interface du routeur 1 **tunnel.100 : 192.168.100.254** de même pour poste 2 **tunnel.200 :192.168.200.254** donc ils passent bien par routeur 1 pour accéder à internet comme le montre les captures ci-dessous.

```

root@amadimk: ~/PEF/INFRA/Projet 70x23
root@amadimk:~/PEF/INFRA/Projet# [poste2] ip r
default via 192.168.200.254 dev poste2-eth0.200
192.168.200.0/24 dev poste2-eth0.200 proto kernel scope link src 192.1
68.200.54
root@amadimk:~/PEF/INFRA/Projet# [poste2] ping -c 2 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=114 time=195 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=114 time=573 ms

--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 195.158/384.411/573.665/189.254 ms
root@amadimk:~/PEF/INFRA/Projet# [poste2]

root@amadimk:~/PEF/INFRA/Projet 70x23
root@amadimk:~/PEF/INFRA/Projet# [poste1] ip r
default via 192.168.100.254 dev poste1-eth0.100
192.168.100.0/24 dev poste1-eth0.100 proto kernel scope link src 192.1
68.100.87
root@amadimk:~/PEF/INFRA/Projet# [poste1] ping -c 2 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=114 time=148 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=114 time=186 ms

--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 148.157/167.319/186.482/19.166 ms
root@amadimk:~/PEF/INFRA/Projet# [poste1]

```

Figure 17: Autorisation d'accès à internet pour poste 1 passant par routeur 1

6.4 Accès Internet optimisation par dnsmasq

Pour pouvoir autoriser internet par le routeur 2 en utilisant **dnsmasq** pour optimiser internet nous avons mis en place un **DHCP Relay** sur le routeur 2 qui permet de relayer les requêtes dhcp des machines de son coté au serveur dhcp se trouvant sur routeur 1 tout en lui spécifiant dans les options des requêtes la route par défaut à utiliser qui est à priori son interface à lui même.

Pour Lancer un **DHCP Relay** avec **dnsmasq** et spécifier une route par défaut à utiliser il suffit de procéder comme suit :

```

$ dnsmasq -d --dhcp-relay=192.168.100.253,192.168.100.254,tunnel
.100 --dhcp-option=3,192.168.100.253

```

Explication : avec l'option **-dhcp-relay** on indique l'adresse de l'interface sur lequel est lancé le dhcp-relay : **192.168.100.253**, l'adresse du serveur dhcp distant : **192.168.100.254** puis avec l'option **-dhcp-option** on indique qu'on veut utiliser comme route par défaut (le 3) l'adresse : **192.168.100.253**

```

root@amadimk:~/PEF/INFRA/Projet 70x23
root@amadimk:~/PEF/INFRA/Projet# [poste1] dhclient poste1-eth0.100
root@amadimk:~/PEF/INFRA/Projet# [poste1] ip r
default via 192.168.100.253 dev poste1-eth0.100
192.168.100.0/24 dev poste1-eth0.100 proto kernel scope link src 192.1
68.100.87
root@amadimk:~/PEF/INFRA/Projet# [poste1] ping -c 3 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=114 time=205 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=114 time=154 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=114 time=74.4 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 74.457/144.653/205.424/53.881 ms
root@amadimk:~/PEF/INFRA/Projet# [poste1]

root@amadimk:~/PEF/INFRA/Projet 70x23
root@amadimk:~/PEF/INFRA/Projet# [route2] dnsmasq -d --dhcp-relay=192.1
68.100.253,192.168.100.254,tunnel --dhcp-option=3,192.168.100.253
dnsmasq: démarré, version 2.79 (taille de cache 150)
dnsmasq: options à la compilation : IPv6 GNU-getopt DBus i18n IDN DHCP
DHCPv6 no-Lua TFTP conntrack ipset auth DNSSEC loop-detect inotify
dnsmasq-dhcp: DHCP relay from 192.168.100.253 to 192.168.100.254 via t
unnel
dnsmasq: Lecture de /etc/resolv.conf
dnsmasq: utilise le serveur de nom 127.0.0.53#53
dnsmasq: lecture /etc/hosts - 7 adresses

```

Figure 18: Mise en oeuvre dhcp relay pour accès internet optimisé

7 Interdiction de trafic entre VLAN 100 et VLAN 200

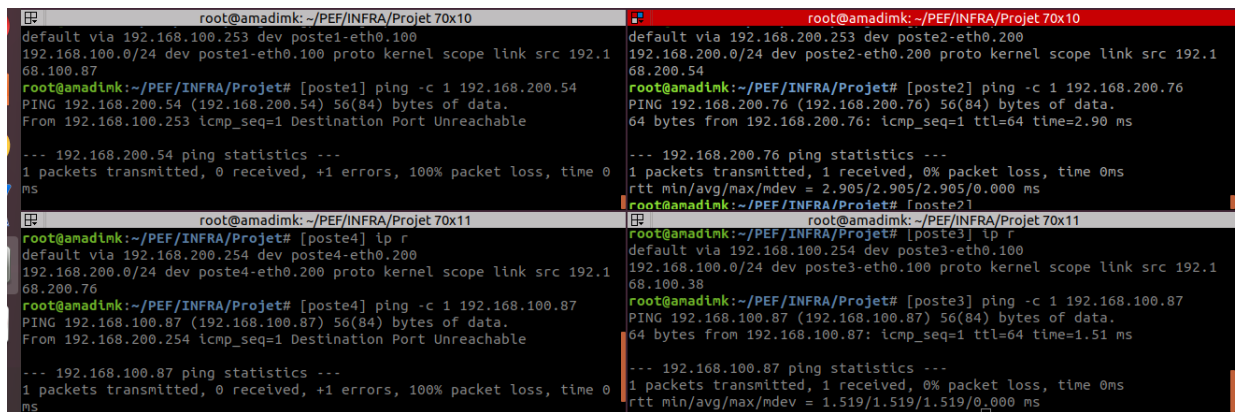
7.1 Interdiction à l'aide d'Iptables

Les règles firewall à exécuter avec iptables sont les suivantes :

```
$ ip netns exec rout1 iptables -A FORWARD -s 192.168.100.0/24 -d 192.168.200.0/24 -j REJECT
$ ip netns exec rout1 iptables -A FORWARD -s 192.168.200.0/24 -d 192.168.100.0/24 -j REJECT

$ ip netns exec rout2 iptables -A FORWARD -s 192.168.100.0/24 -d 192.168.200.0/24 -j REJECT
$ ip netns exec rout2 iptables -A FORWARD -s 192.168.200.0/24 -d 192.168.100.0/24 -j REJECT
```

Le fichier permettant de faire le build de ces règles sur nos routeurs 1 et 2 est : **build_iptables_deny_vlan100_vlan200**. Ainsi, comme nous pouvons le constater sur la capture suivante les postes se trouvant sur le même vlan arrivent à communiquer par ping et n'arrivent pas à faire le ping si le vlan est différent.



The figure consists of four terminal screenshots arranged in a 2x2 grid, showing network configurations and ping results for two different network namespaces (VLAN 100 and VLAN 200).

- Top Left:** Shows the configuration of the first router (rout1) in the 'exec' namespace. It sets up iptables rules to deny traffic between 192.168.100.0/24 and 192.168.200.0/24. The output shows the rules being added successfully.
- Top Right:** Shows the configuration of the second router (rout2) in the 'exec' namespace. It sets up iptables rules to deny traffic between 192.168.100.0/24 and 192.168.200.0/24. The output shows the rules being added successfully.
- Bottom Left:** Shows a ping test from the first router (rout1) to the second router (rout2). The ping fails, indicating that the traffic is blocked by the firewall rules.
- Bottom Right:** Shows a ping test from the second router (rout2) to the first router (rout1). The ping fails, indicating that the traffic is blocked by the firewall rules.

Figure 19: Interdiction communication VLAN 100 et VLAN 200 Iptables

On constate sur la capture que le ping du poste 1 vers le poste 2 ne marche pas, que le ping du poste 3 vers le poste 1 marche bien, aussi le ping du poste 4 vers le poste 3 ne marche pas mais par contre celui du poste 2 vers poste 4 marche bien.

7.2 Interdiction à l'aide de Policy Routing

Pour l'interdiction avec Policy routing, nous avons ajouté deux nouvelles tables de routage **vlan100** et **vlan200** en éditant le fichier `/etc/iproute2/rt_tables` ensuite nous avons exécuté les commandes suivantes :

```
#rout1
$ ip netns exec rout1 ip rule add from 192.168.100.0/24 lookup
  vlan100
$ ip netns exec rout1 ip rule add from 192.168.200.0/24 lookup
  vlan200

#rout2
$ ip netns exec rout2 ip rule add from 192.168.100.0/24 lookup
  vlan100
$ ip netns exec rout2 ip rule add from 192.168.200.0/24 lookup
  vlan200

# Ici nous avons preferez une interdiction administrative de
  communication entre les 2 vlans : prohibit

#rout1
$ ip netns exec rout1 ip route add prohibit 192.168.200.0/24
  table vlan100
$ ip netns exec rout1 ip route add prohibit 192.168.100.0/24
  table vlan200

#rout2
$ ip netns exec rout2 ip route add prohibit 192.168.200.0/24
  table vlan100
$ ip netns exec rout2 ip route add prohibit 192.168.100.0/24
  table vlan200
```

Le fichier permettant de faire le build de ces règles sur nos routeurs 1 et 2 est : **build_policy_routing_deny_vlan100_vlan200**.

<pre>root@amadink: ~/PEF/INFRA/Projet 70x10 root@amadink:~/PEF/INFRA/Projet# [poste1] ip r default via 192.168.100.254 dev poste1-eth0.100 192.168.100.0/24 dev poste1-eth0.100 proto kernel scope link src 192.168.100.7 root@amadink:~/PEF/INFRA/Projet# [poste1] ping -c 1 192.168.200.76 PING 192.168.200.76 (192.168.200.76) 56(84) bytes of data. From 192.168.100.254 icmp_seq=1 Packet filtered --- 192.168.200.76 ping statistics --- 1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0 ms</pre>	<pre>root@amadink: ~/PEF/INFRA/Projet 70x10 root@amadink:~/PEF/INFRA/Projet# [poste2] ip r default via 192.168.200.254 dev poste2-eth0.200 192.168.200.0/24 dev poste2-eth0.200 proto kernel scope link src 192.168.200.54 root@amadink:~/PEF/INFRA/Projet# [poste2] ping -c 1 192.168.100.43 PING 192.168.100.43 (192.168.100.43) 56(84) bytes of data. From 192.168.200.254 icmp_seq=1 Packet filtered --- 192.168.100.43 ping statistics --- 1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0 ms</pre>
<pre>root@amadink: ~/PEF/INFRA/Projet 70x11 root@amadink:~/PEF/INFRA/Projet# [poste3] ip r default via 192.168.100.254 dev poste3-eth0.100 192.168.100.0/24 dev poste3-eth0.100 proto kernel scope link src 192.168.100.43 root@amadink:~/PEF/INFRA/Projet# [poste3] ping -c 1 192.168.100.7 PING 192.168.100.7 (192.168.100.7) 56(84) bytes of data. 64 bytes from 192.168.100.7: icmp_seq=1 ttl=64 time=3.27 ms --- 192.168.100.7 ping statistics --- 1 packets transmitted, 1 received, 0% packet loss, time 0ms rtt min/avg/max/mdev = 3.275/3.275/3.275/0.000 ms</pre>	<pre>root@amadink: ~/PEF/INFRA/Projet 70x11 root@amadink:~/PEF/INFRA/Projet# [poste4] ip r default via 192.168.200.254 dev poste4-eth0.200 192.168.200.0/24 dev poste4-eth0.200 proto kernel scope link src 192.168.200.76 root@amadink:~/PEF/INFRA/Projet# [poste4] ping -c 1 192.168.200.54 PING 192.168.200.54 (192.168.200.54) 56(84) bytes of data. 64 bytes from 192.168.200.54: icmp_seq=1 ttl=64 time=1.64 ms --- 192.168.200.54 ping statistics --- 1 packets transmitted, 1 received, 0% packet loss, time 0ms rtt min/avg/max/mdev = 1.644/1.644/1.644/0.000 ms</pre>

Figure 20: Interdiction communication VLAN 100 et VLAN 200 Policy Routing

On constate sur la capture que le ping du poste 1 vers le poste 4 ne marche pas avec **packet filtered**, que le ping du poste 3 vers le poste 1 marche bien, aussi le ping du poste 4 vers le poste 2 marche bien mais par contre celui du poste 2 vers poste 3 ne marche pas.