ASSIGNMENT 6 – LAB 8: OVERLAY NETWORKS

Main Topics: Overlay networks, why is tunnel encapsulation important, Wireguard and VPN in general

- Basic topology:

(entra da multipass un terminale)
sudo apt install docker.io
sudo docker run --privileged -it --name r1 ubuntu /bin/bash (I had to apply this instead of an "exec" because of problems with the virtual machine and repositories)
Everytime my VM doesn't know the packages so I have always to install them (for ex ip with **sudo apt update apt install -y iproute2**)
root@5c5dd4638b3e:/# ip netns add R1
ip link add veth-r1 type veth peer name veth-r2
ip link set veth-r1 netns R1
ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: gre0@NONE: <NOARP> mtu 1476 qdisc noop state DOWN mode DEFAULT group default qlen 1000
link/gre 0.0.0.0 brd 0.0.0.0
3: gretap0@NONE: <BROADCAST,MULTICAST> mtu 1462 qdisc noop state DOWN mode DEFAULT group default qlen 1000
link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
4: erspan0@NONE: <BROADCAST,MULTICAST> mtu 1450 qdisc noop state DOWN mode DEFAULT group default qlen 1000
link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
5: eth0@if49: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
link/ether f2:53:53:02:34:b6 brd ff:ff:ff:ff:ff:ff link-netnsid 0
6: veth-r2@if7: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
link/ether 96:56:c7:e2:93:b6 brd ff:ff:ff:ff:ff:ff link-netns R1
ip netns add R2
After having added the two routers I've created the ethernet cables (virtual) and moved the veth-r1 in the correspondent namespace and the same for R2:
ip link set veth-r2 netns R2
ip link add veth-r2 type veth peer name veth-r1
I've assigned the Ip addresses and activated the interfaces.

ip netns exec R1 ip addr add 10.0.0.1/24 dev veth-r1

ip netns exec R2 ip addr add 10.0.0.2/24 dev veth-r2

ip netns exec R1 ip link set veth-r1 up

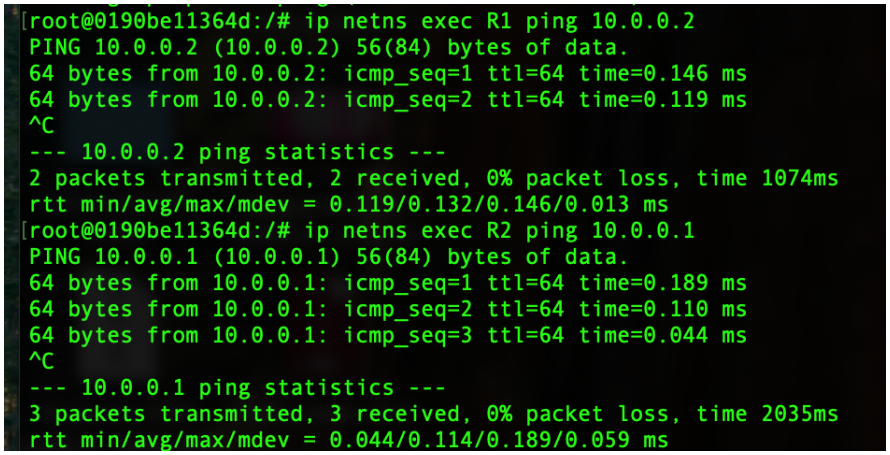
ip netns exec R2 ip link set veth-r2 up

(Also here I had the same problema s before of not having ping: **apt update apt install -y iputils-ping)**

Then I have tested the connectivity between all the entities:

ip netns exec R1 ping 10.0.0.2

ip netns exec R2 ping 10.0.0.1

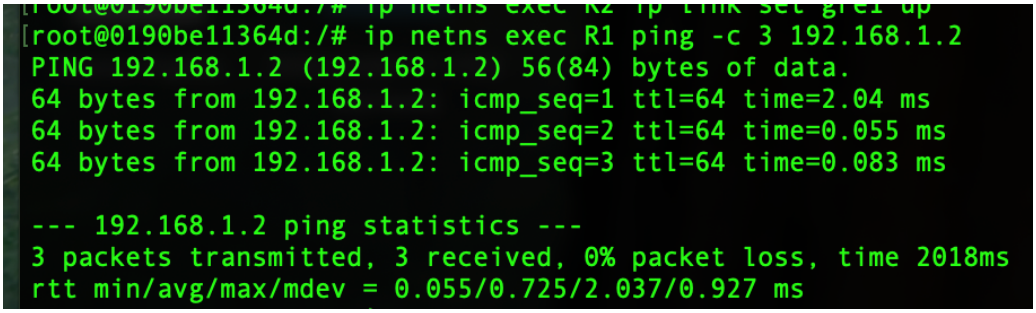


- GRE Tunnel:

`apt install kmod -y to install modprobe and modprobe ip_gre to enable the kernel.`

GRE stands for Generic Routing Encapsulation. If the veth pair represents the underlay path, the GRE is the core of the overlay concept.

ip netns exec R1 ip tunnel add gre1 mode gre remote 10.0.0.2 local 10.0.0.1 ttl 63
ip netns exec R1 ip addr add 192.168.1.1/24 dev gre1
ip netns exec R1 ip link set gre1 up
ip netns exec R2 ip tunnel add gre1 mode gre remote 10.0.0.1 local 10.0.0.2 ttl 63
ip netns exec R2 ip addr add 192.168.1.2/24 dev gre1
ip netns exec R2 ip link set gre1 up
ip netns exec R1 ping -c 3 192.168.1.2



- VXLAN:

VXLAN (Virtual Extensible LAN) encapsulates Layer 2 Ethernet frames into Layer 4 UDP packets.

Creation of with the VNI + destination port indicated is that of the UDP

ip netns exec R1 ip link add vxlan0 type vxlan id 100 local 10.0.0.1 remote 10.0.0.2 nolearning dstport 4789

ip netns exec R1 ip addr add 172.16.1.1/24 dev vxlan0

ip netns exec R1 ip link set vxlan0 up

I have prevented the interface from learning MAC addresses automatically in order to increase the security of the traffic from uknown MAC addresses and to prevent from possible external attacks.

ip netns exec R2 ip link add vxlan0 type vxlan id 100 local 10.0.0.2 remote 10.0.0.1 nolearning dstport 4789

ip netns exec R2 ip addr add 172.16.1.2/24 dev vxlan0

ip netns exec R2 ip link set vxlan0 up

ip netns exec R1 ping -c 3 172.16.1.2

- WireGuard Setup:

WireGuard is a modern, high-performance VPN that uses public-key cryptography.

To install it: **apt install wireguard**

To keep in mind: **51820** is the **standard default port** for WireGuard

Generation of the private key for R1 and R2:

wg genkey | tee r1_private.key

wg genkey | tee r2_private.key

ip netns exec R1 ip link add dev wg0 type wireguard

ip netns exec R1 wg set wg0 private-key r1_private.key listen-port 51820

ip netns exec R1 ip addr add 10.10.10.1/24 dev wg0

ip netns exec R1 ip link set wg0 up

ip netns exec R2 ip link add dev wg0 type wireguard

ip netns exec R2 wg set wg0 private-key r2_private.key listen-port 51820

ip netns exec R2 ip addr add 10.10.10.2/24 dev wg0

ip netns exec R2 ip link set wg0 up

cat r1_private.key | wg pubkey

cat r2_private.key | wg pubkey

Peers linked by specifying the public key of the other end and the endpoint IP:

ip netns exec R1 wg set wg0 peer *result from cat r2_private* endpoint 10.0.0.2:51820 allowed-ips 10.10.10.0/24

ip netns exec R2 wg set wg0 peer *result from cat r1_private* endpoint 10.0.0.1:51820 allowed-ips 10.10.10.0/24

ip netns exec R1 wg

ip netns exec R2 wg

ip netns exec R1 ping 10.10.10.2

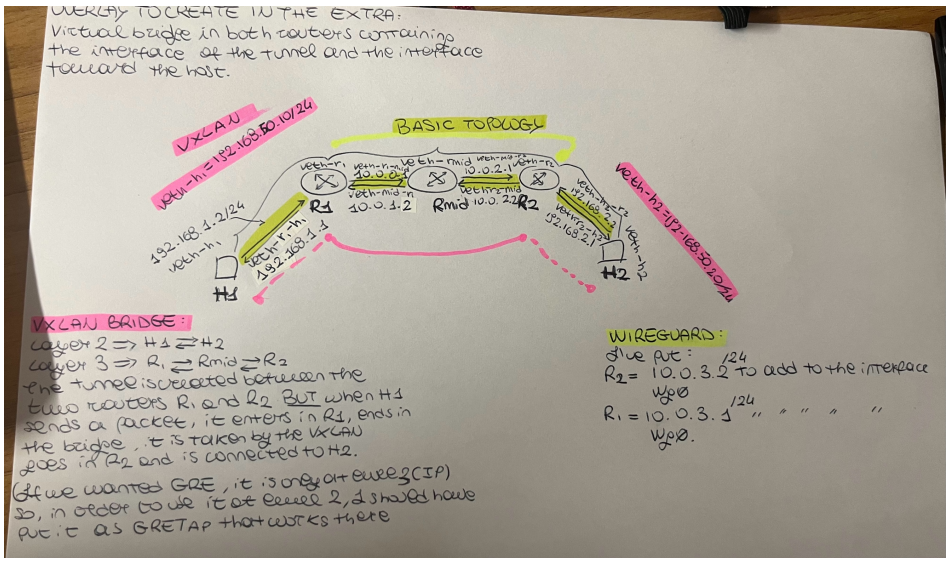
ip netns exec R2 ping 10.10.10.1

```
root@0190be11364d:/# ip netns exec R1 wg
interface: wg0
  public key: n3t9PgWrPxN+uA4agKcYPXhsbVENYaPkxKvaxPNu9Ho=
  private key: (hidden)
  listening port: 57874

peer: bbHtXQiulOSU7JdALmaYAsOqSan11Iwj1XZ6Z04+x0M=
  endpoint: 10.0.0.2:51820
  allowed ips: 10.10.10.0/24
root@0190be11364d:/# ip netns exec R2 wg
interface: wg0
  public key: bbHtXQiulOSU7JdALmaYAsOqSan11Iwj1XZ6Z04+x0M=
  private key: (hidden)
  listening port: 36101

peer: n3t9PgWrPxN+uA4agKcYPXhsbVENYaPkxKvaxPNu9Ho=
  endpoint: 10.0.0.1:51820
  allowed ips: 10.10.10.0/24
root@0190be11364d:/# ip netns exec R2 wg set wg0 listen-port 51820
root@0190be11364d:/# ip netns exec R1 ping 10.10.10.2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=3.40 ms
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=0.287 ms
64 bytes from 10.10.10.2: icmp_seq=3 ttl=64 time=0.077 ms
^C
--- 10.10.10.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2073ms
rtt min/avg/max/mdev = 0.077/1.253/3.395/1.517 ms
```

EXTRA: ex. Let's add the middle router and the two hosts:



 # We clean the previous environment

ip -all netns delete 2>/dev/null

rm *.key 2>/dev/null

 # Creation of the entities (namespaces)

ip netns add R1

ip netns add R2

ip netns add Rmid

ip netns add H1

ip netns add H2

- **Basic topology:**

ip netns add Rmid

ip netns add H1

ip netns add H2

ip link add veth-h1 type veth peer name veth-r1-h1

ip link set veth-h1 netns H1

ip link set veth-r1-h1 netns R1

ip link add veth-r1-mid type veth peer name veth-mid-r1

ip link set veth-r1-mid netns R1

ip link set veth-mid-r1 netns Rmid

ip link add veth-mid-r2 type veth peer name veth-r2-mid

ip link set veth-mid-r2 netns Rmid

ip link set veth-r2-mid netns R2

ip link add veth-r2-h2 type veth peer name veth-h2

ip link set veth-r2-h2 netns R2

ip link set veth-h2 netns H2

ip netns exec H1 ip addr add 192.168.1.2/24 dev veth-h1

ip netns exec R1 ip addr add 192.168.1.1/24 dev veth-r1-h1

ip netns exec H2 ip addr add 192.168.2.2/24 dev veth-h2

ip netns exec R2 ip addr add 192.168.2.1/24 dev veth-r2-h2

ip netns exec H1 ip route add default via 192.168.1.1

ip netns exec H2 ip route add default via 192.168.2.1

ip netns exec R1 sysctl -w net.ipv4.ip_forward=1

ip netns exec Rmid sysctl -w net.ipv4.ip_forward=1

ip netns exec R2 sysctl -w net.ipv4.ip_forward=1

ip netns exec R1 ip addr add 10.0.1.1/24 dev veth-r1-mid

ip netns exec R1 ip link set veth-r1-mid up

ip netns exec Rmid ip addr add 10.0.1.2/24 dev veth-mid-r1

ip netns exec Rmid ip addr add 10.0.2.1/24 dev veth-mid-r2

ip netns exec Rmid ip link set veth-mid-r1 up

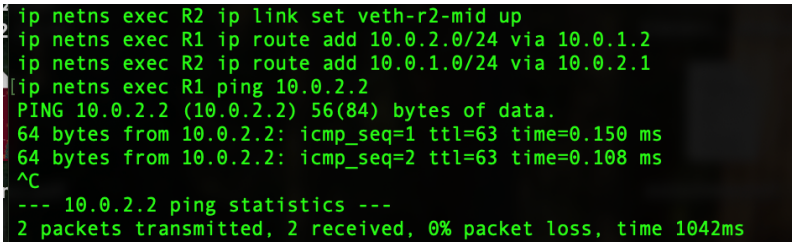
ip netns exec Rmid ip link set veth-mid-r2 up

ip netns exec R2 ip addr add 10.0.2.2/24 dev veth-r2-mid

ip netns exec R2 ip link set veth-r2-mid up

ip netns exec R1 ip route add 10.0.2.0/24 via 10.0.1.2

ip netns exec R2 ip route add 10.0.1.0/24 via 10.0.2.1



#In order to let H1 and H2 talk to each other...

ip netns exec H1 ip addr add 192.168.1.2/24 dev veth-h1

ip netns exec H1 ip link set veth-h1 up

ip netns exec R1 ip addr add 192.168.1.1/24 dev veth-r1-h1

ip netns exec R1 ip link set veth-r1-h1 up

ip netns exec H2 ip addr add 192.168.2.2/24 dev veth-h2

ip netns exec H2 ip link set veth-h2 up

ip netns exec R2 ip addr add 192.168.2.1/24 dev veth-r2-h2

ip netns exec R2 ip link set veth-r2-h2 up

ip netns exec H1 ip route add default via 192.168.1.1

ip netns exec H2 ip route add default via 192.168.2.1

ip netns exec R1 sysctl -w net.ipv4.ip_forward=1

ip netns exec Rmid sysctl -w net.ipv4.ip_forward=1

ip netns exec R2 sysctl -w net.ipv4.ip_forward=1

ip netns exec R1 ip route add 192.168.2.0/24 via 10.0.1.2

ip netns exec R2 ip route add 192.168.1.0/24 via 10.0.2.1

```
[root@51124788bae5:/# ip netns exec H2 ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=61 time=0.354 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=61 time=0.130 ms
^C
--- 192.168.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1056ms
rtt min/avg/max/mdev = 0.130/0.242/0.354/0.112 ms
[root@51124788bae5:/# ip netns exec H1 ping 192.168.2.2
PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data.
64 bytes from 192.168.2.2: icmp_seq=1 ttl=61 time=0.146 ms
64 bytes from 192.168.2.2: icmp_seq=2 ttl=61 time=0.192 ms
^C
--- 192.168.2.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1026ms
rtt min/avg/max/mdev = 0.146/0.169/0.192/0.023 ms
root@51124788bae5:/#
```

- **VXLAN:**

ip netns exec R1 ip link add vx-extra type vxlan id 500 local 10.0.1.1 remote 10.0.2.2 dstport 4789

ip netns exec R1 ip link set vx-extra up

ip netns exec R1 ip link add br0 type bridge

ip netns exec R1 ip link set veth-r1-h1 master br0

ip netns exec R1 ip link set vx-extra master br0

ip netns exec R1 ip link set br0 up

ip netns exec R2 ip link add vx-extra type vxlan id 500 local 10.0.2.2 remote 10.0.1.1 dstport 4789

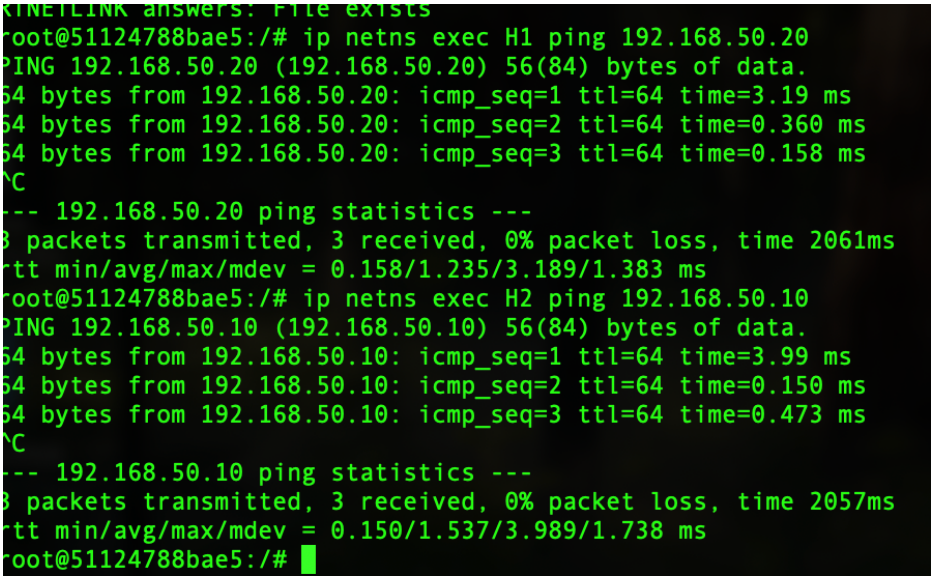
ip netns exec R2 ip link set vx-extra up

ip netns exec R2 ip link add br0 type bridge

ip netns exec R2 ip link set veth-r2-h2 master br0

ip netns exec R2 ip link set vx-extra master br0

ip netns exec R2 ip link set br0 up



-GRE (eventually, GRETAP)

ip netns exec R1 ip link add gre-extra type gretap local 10.0.1.1 remote 10.0.2.2 ttl 63

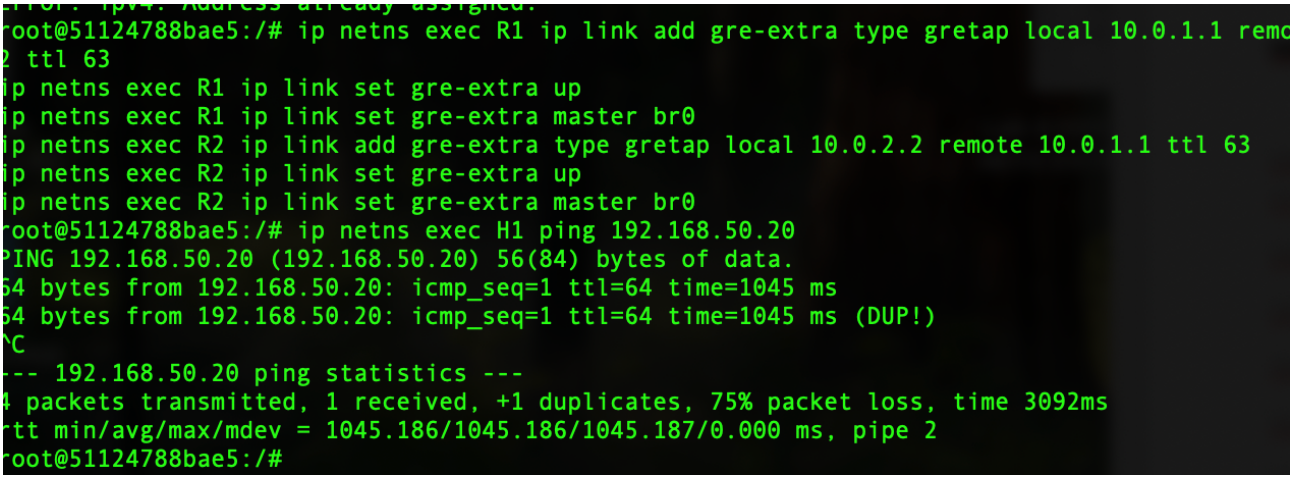
ip netns exec R1 ip link set gre-extra up

ip netns exec R1 ip link set gre-extra master br0

ip netns exec R2 ip link add gre-extra type gretap local 10.0.2.2 remote 10.0.1.1 ttl 63

ip netns exec R2 ip link set gre-extra up

ip netns exec R2 ip link set gre-extra master br0



#WIREGUARD PART:

ip netns exec R1 ip link add dev wg0 type wireguard

ip netns exec R1 bash -c "wg genkey | tee private_r1.key | wg pubkey > public_r1.key"

ip netns exec R1 wg set wg0 private-key private_r1.key

ip netns exec R1 ip addr add 10.0.3.1/24 dev wg0

ip netns exec R1 ip link set wg0 up

ip netns exec R2 ip link add dev wg0 type wireguard

ip netns exec R2 bash -c "wg genkey | tee private_r2.key | wg pubkey > public_r2.key"

ip netns exec R2 wg set wg0 private-key private_r2.key

ip netns exec R2 ip addr add 10.0.3.2/24 dev wg0

ip netns exec R2 ip link set wg0 up

cat private_r1.key

cat public_r1.key

cat private_r2.key

cat public_r2.key

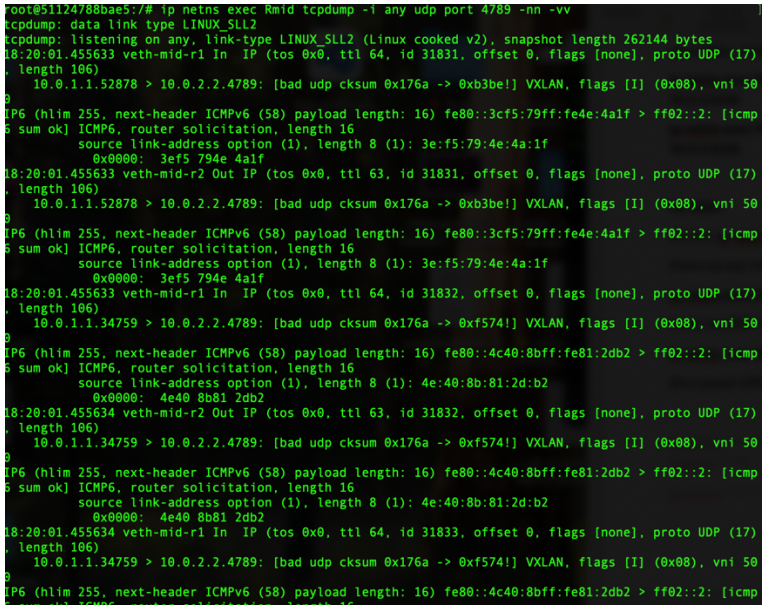
ip netns exec R1 wg set wg0 peer <R2 PUB KEY> endpoint 10.0.2.2:51820 allowed-ips 10.0.3.0/24

ip netns exec R2 wg set wg0 <R1 PUB KEY> peer endpoint 10.0.1.1:51820 allowed-ips 10.0.3.0/24

TO TEST:

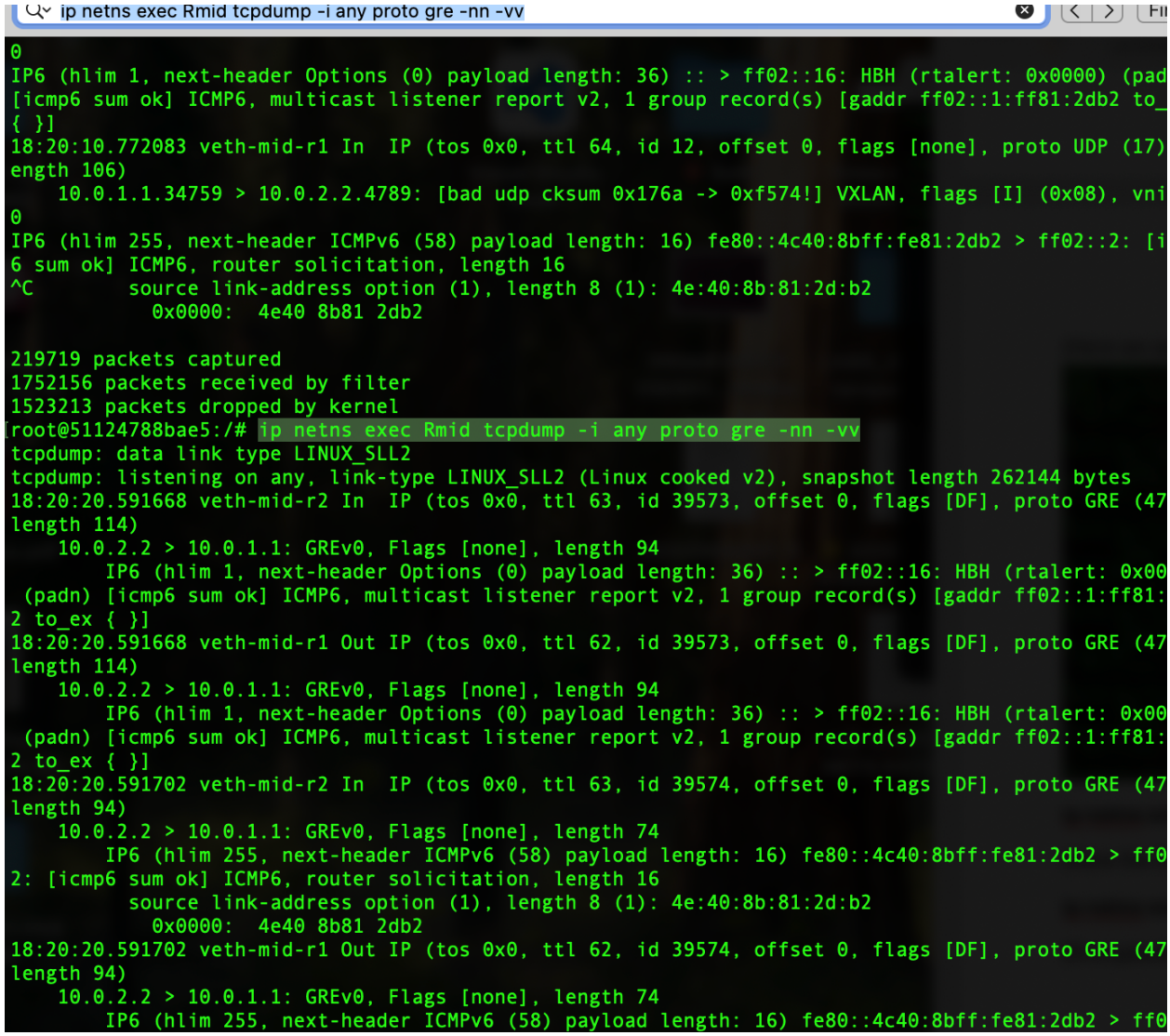
VXLAN: **ip netns exec Rmid tcpdump -i any udp port 4789 -nn -vv**

(Here we see the UDP header and the internal Ethernet frame.)

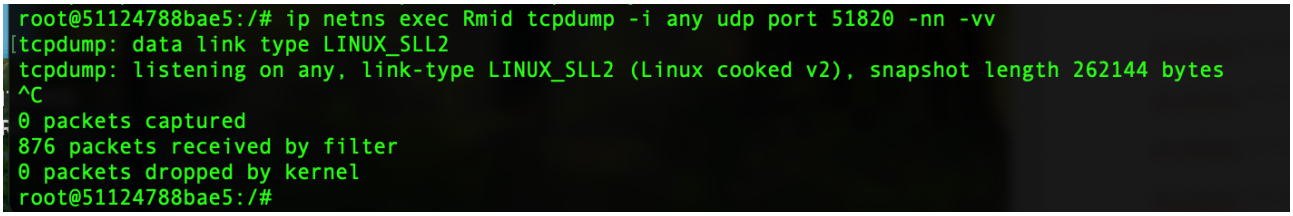


ip netns exec Rmid tcpdump -i any proto gre -nn -vv

(Here the GRE header)



ip netns exec Rmid tcpdump -i any udp port 51820 -nn -vv



(Encrypted UDP traffic)

N.B. the flags nn and vv stands for numeric(in order to not translate numbers in names (ex. We don't want the original IPs in names of the hosts) and verbose (to check the level of detail in the packet like TTL or ID).