GRE

Протокол GRE создавался компанией Cisco Systems для организации сетевых тунелей. И хотя в настоящее время GRE теряет актуальность, так как не поддерживает шифрование и не работает через NAT, в случае необходимости быстрой организации туннеля между двумя локальными сетями именно GRE будет в числе самых простых вариантов.

Два маршрутизатора (Net-R0 и Net-R2) на базе Linux CentOS 7 с такими вводными:

Net-R0:

WAN enp0s3 192.168.113.63
LAN enp0s8 10.0.0.1
GRE 172.17.254.1

Net-R2:

WAN enp0s3 192.168.113.65
LAN enp0s8 172.16.8.1
GRE 172.17.254.2

Поднять тонель и получить доступ к внутренним сетям.

```
# sysctl net.ipv4.ip_forward=1
```

Net-R0 (Host1)

```
# cat ifcfg-gre1
DEVICE=gre1
BOOTPROTO=none
ONBOOT=no
TYPE=GRE

## Addr Srv Net-R0
MY_OUTER_IPADDR=192.168.113.63
MY_INNER_IPADDR=172.17.254.1

PEER_OUTER_IPADDR=192.168.113.65
```

```
PEER_INNER_IPADDR=172.17.254.2
```

```
# cat route-gre1
172.16.8.0/24 via 172.17.254.2 dev gre1
```

Net-R2 (Host2)

```
# cat ifcfg-gre1
DEVICE=gre1
BOOTPROTO=none
ONBOOT=no
TYPE=GRE

## Addr Srv Net-R2
MY_OUTER_IPADDR=192.168.113.65
## Addr Srv in a tonnel
MY_INNER_IPADDR=172.17.254.2

# Addr Peer (other side Net-R0)
PEER_OUTER_IPADDR=192.168.113.63
PEER_INNER_IPADDR=172.17.254.1
```

```
# cat route-gre1
10.0.0.0/24 via 172.17.254.1 dev gre1
```

Firewall

```
# firewall-cmd --permanent --direct --add-rule ipv4 filter
INPUT 0 -p gre -j ACCEPT
# firewall-cmd --permanent --direct --add-rule ipv6 filter
INPUT 0 -p gre -j ACCEPT
# firewall-cmd --reload
```

Iptables

```
# iptables -I INPUT -p gre -j ACCEPT
```

Проверяем:

```
[root@Net-R2]# ping 10.0.0.1
```

```
[root@net-r0]# tcpdump -envvn proto gre
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet),
capture size 262144 bytes
```

```
14:37:45.776948  08:00:27:5b:03:19  >  08:00:27:5c:5e:08,
ethertype IPv4 (0x0800), length 122: (tos 0x0, ttl 64, id 761,
offset 0, flags [DF], proto GRE (47), length 108)
   192.168.113.65 > 192.168.113.63: GREv0, Flags [none], proto
IPv4 (0x0800), length 88
    (tos 0x0, ttl 64, id 24605, offset 0, flags [DF], proto
ICMP (1), length 84)
   172.17.254.2 > 10.0.0.2: ICMP echo request, id 9655, seq 1,
length 64
14:37:46.957099  08:00:27:5b:03:19  >  08:00:27:5c:5e:08,
ethertype IPv4 (0x0800), length 122: (tos 0x0, ttl 64, id
1554, offset 0, flags [DF], proto GRE (47), length 108)
   192.168.113.65 > 192.168.113.63: GREv0, Flags [none], proto
IPv4 (0x0800), length 88
    (tos 0x0, ttl 64, id 24723, offset 0, flags [DF], proto
ICMP (1), length 84)
   172.17.254.2 > 10.0.0.2: ICMP echo request, id 9655, seq 2,
length 64
```

Краткая шпаргалка:

```
HOST1: ip link add grelan type gretap  local <IP1> remote
<IP2>
HOST1: ip link set grelan up
HOST1: iptables -I INPUT -p gre -s <IP2> -j ACCEPT
HOST2: ip link add grelan type gretap local <IP2> remote <IP1>
HOST2: ip link set grelan up
HOST2: iptables -I INPUT -p gre -s <IP1> -j ACCEPT
```