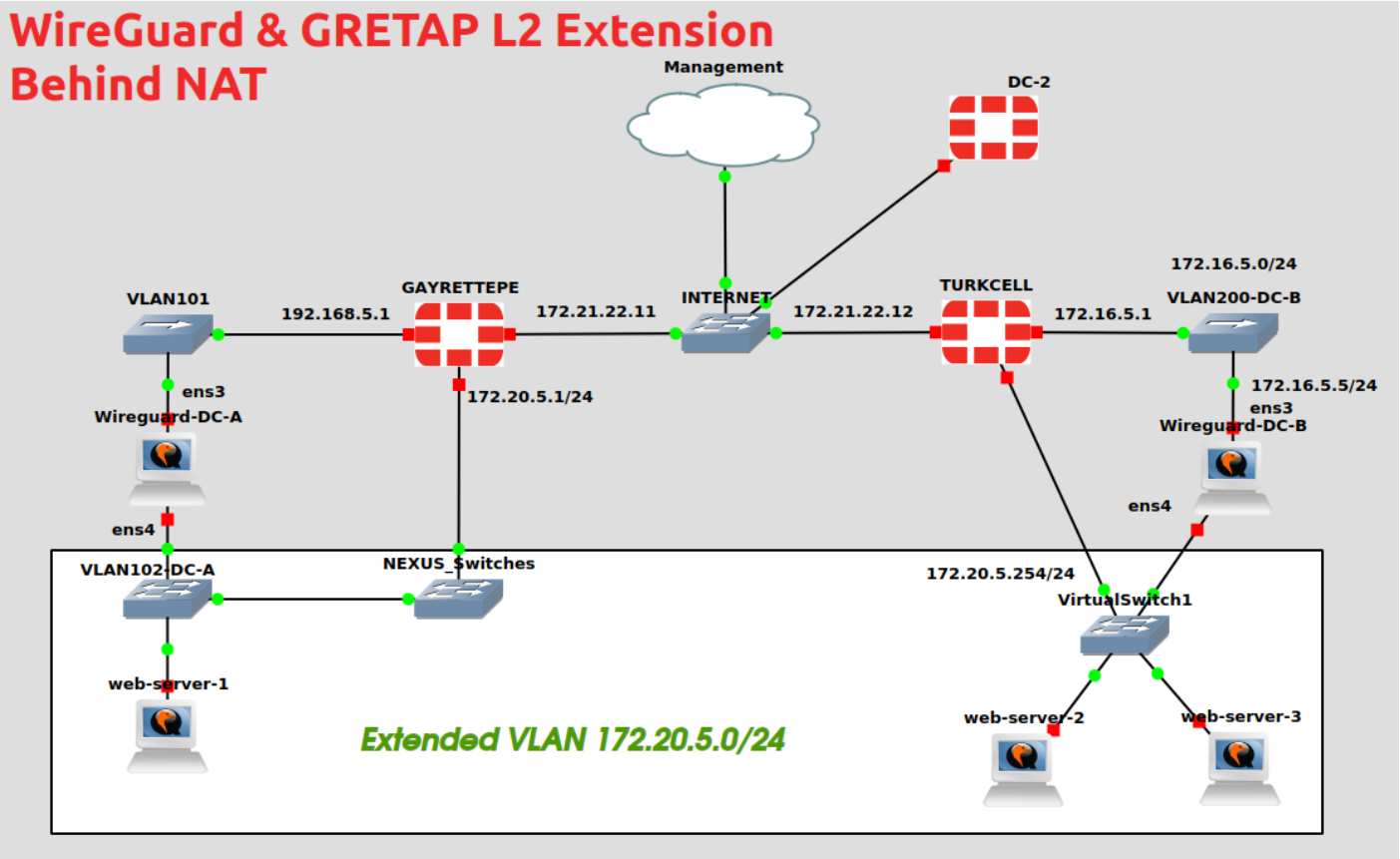
Extending l2 networks using wireguard & gretap



At some cases, you might need to extend a vlan over l3 links, and for some reason while;

- you don't want to spend money on devices
- change routing/interface configurations on firewalls.
- No vxlan capable devices in use (i know, there vpls, l2tpv3 and some other solutions but hey it's linux so why not ?)

Then here is your solution, with a simple dnat entry you can extend your vlan.

In my case, customer was in need to backup their vm's using VMWare's replication to their DR center and whenever they needed, they do like to use the DR as Active data center. And changing configs on a disaster situation was the least they want to deal with.

Of course this is only a half of a complete DR solution may less, but restoring a services directly from DR was the requirement as i've been told.

Requirements:

- 2 ubuntu 18.04 machines on both data centers
- for each ubuntu vm 1 interface for l3 tunnel as underlay and one interface to use as tap to extend vlan
- 2 cores were quite powerfull for 200 mbits of througput
- 1 gb ram per machine
- Allowing forged ethernet frames on both tap ports
- 1 cup of coffee and a slice of good chestnut cake

Some caveats ;

- good cpu = good throughput, and the opposite
- do not loop !
- be carefull with the Xstp or you might have a lame root bridge
- My config was interface per vlan, so somone might have a trunk solution.
- No HA solution, those machines needs to be monitored closely, use observium and it's agent

install wireguard

To install wireguard vpn on ubuntu 18.04

```
#sudo apt update
#sudo apt-get install libmnl-dev libelf-dev linux-headers-$(uname -r) build-essential pkg-config
#sudo apt install wireguard
```

After installation you'll need to restart operating system and check for installed module, output should be as shown below.

```
#lsmod| grep wireguard
udp_tunnel□16384 1 wireguard
ip6_udp_tunnel 16384 1 wireguard
```

generete keys

Wireguard need public and private keys to operate, there is a tool called wg which can generate them as shown below

```
#wg genkey | sudo tee /etc/wireguard/privatekey | wg pubkey | sudo tee /etc/wireguard/publickey
```

You can find the keys at /etc/wireguard

create wireguard0.conf

You'll need to create a file called wireguard0.conf at /etc/wireguard

Contents of the file should be like this ;

For initiator:

```
[interface]
private_key= the_private_key_you_generated
address= 10.10.10.1/24 #ip address of the wireguard0

[peer]
PublicKey = #he public key that you generated on the responder host
AllowedIPs = 10.10.10.0/24 #and the other networks that you'd like route through
EndPoint = internet_address:port
PersistenKeepAlive = 15 #seconds
```

For responder:

```
[interface]
private_key= the_private_key_you_generated
address= 10.10.10.2/24 #ip address of the wireguard0
listenPort = 4900

[peer]
PublicKey = #he public key that you generated on the initiator host
AllowedIPs = 10.10.10.0/24 #nd the other networks that you'd like route through
PersistenKeepAlive = 15 #seconds
```

initiate tunnel

to start the tunnel

```
wg-quick up wireguard0
```

to check the status

```
wg show wireguard0
interface: wireguard0
  public key: +/1R3JqLKlszbaGUSBtckoxNOMuSvLYKUCl03ShoFw8=
  private key: (hidden)
  listening port: 4900

peer: 2f/RmbuvKtR/L2ZFlQBHsVGkTXkA6d1pJO1ay5EjwSQ=
  endpoint: 172.21.23.111:49792
  allowed ips: 10.10.10.0/24, 192.168.5.0/24
  latest handshake: 1 minute, 31 seconds ago
  transfer: 19.10 MiB received, 11.95 MiB sent
```

install bridge-utils

To install bridge utils

```
#apt install bridge-utils
```

enable br_netfilter

to enable

```
#modprobe br_netfilter
```

to keep loading on boot

```
#sudo sh -c 'echo "br_netfilter" > /etc/modules-load.d/br_netfilter.conf'
#cat net.bridge.bridge-nf-call-ip6tables = 1 >> /etc/sysctl.d/bridge.conf
```

enable routing

To enable routing on the fly

```
sysctl -w net.ipv4.ip_forward=1
```

to make it permanent, add lines below to /etc/sysctl.conf ;

```
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

fix mss on exit interface

When you are going to use ip tunnel for underlay and gretap for overlay, there will be some serious mss/mtu size problems to fix that, we need the br_netfilter module that we installed before and a special chain to limit the mss size to the max mtu of interface

```
#iptables -I FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu
```

We specially use -I to put it at the top of the forward chain.

configure gretap interface

For the initiator:

```
ip link add gretap0 type gretap local 172.16.5.2 remote 192.168.5.2
```

For responder

```
ip link add gretap0 type gretap local 192.168.5.2 remote 172.16.5.2
```

configure bridge

Now you will add the interfaces to bridge

```
#brctl addbr br0
```

add interfaces to bridge br0

```
#brctl addif br0 ens4
#brctl addif br0 gretap0
```

Bring everything up

```
#wp-quick up wireguard0
#ip link set up dev br0
#ip link set up dev gretap0
#ip link et up dev ens4
```

tests

do pings i've checked that dhcp is working,

what needs to be done are ;

- Multicast packets
- the situation about protocols like vrrp and hsrp
- The effect of broadcast packets to cpu

Revision #1
Created 2022-11-09 12:01:23 UTC by Mesut Bayrak
Updated 2022-11-09 12:02:02 UTC by Mesut Bayrak