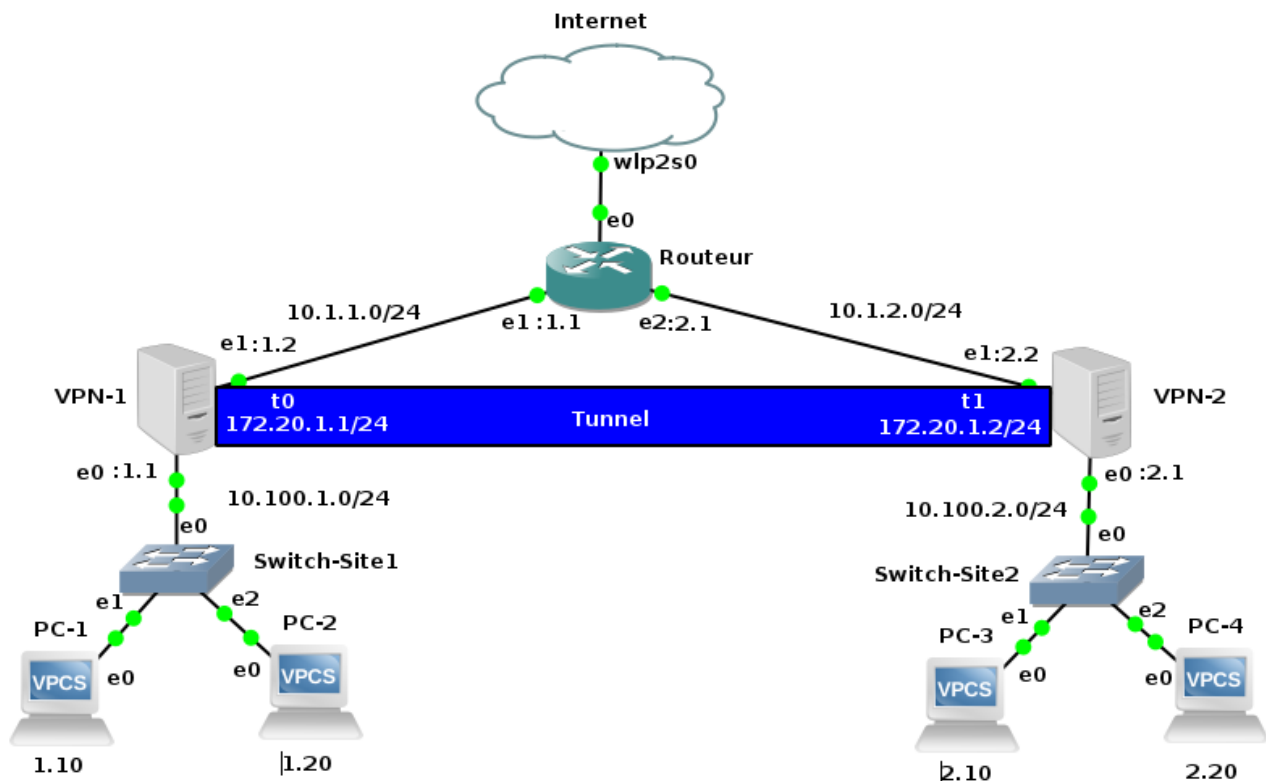


Fiche TP N°1 : Technologies Réseaux - Solution

Partie 2 : Réseaux Privés Virtuels (VPN)

1.VPN site-à-site avec GRE

Nous considérons le schéma réseau d'une entreprise avec deux sites reliés par un tunnel VPN représenté ci-dessous.



Remarque :

Le login et mot de passe pour les serveurs VPN sont :

Login : gns3

Password : gns3

a) Câbler le schéma réseau ci-dessus. Ensuite, configurez :

- Les clients de chaque réseau de l'entreprise (PC1, PC2, PC3, PC4). **(1point)**

PC 1 :

ip 10.100.1.10/24 10.100.1.1

save pc1

PC 2 :

ip 10.100.1.20/24 10.100.1.1

save pc2

PC 3 :

ip 10.100.2.10/24 10.100.2.1

save pc3

PC 4 :

ip 10.100.2.20/24 10.100.2.1

save pc4

- les serveurs VPN : VPN1 et VPN2 (interfaces, routage, NAT) : **(3 points)**

VPN1 :

sudo ip address add 10.100.1.1/24 dev eth0

sudo ip link set dev eth0 up

sudo ip address add 10.1.1.2/24 dev eth1

sudo ip link set dev eth1 up

sudo ip route add default via 10.1.1.1 dev eth1

sysctl -w net.ipv4.ip_forward=1

sudo iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE

VPN2 :

sudo ip address add 10.100.2.1/24 dev eth0

sudo ip link set dev eth0 up

sudo ip address add 10.1.2.2/24 dev eth1

sudo ip link set dev eth1 up

sudo ip route add default via 10.1.2.1 dev eth1

sysctl -w net.ipv4.ip_forward=1

sudo iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE

- Les interfaces du routeur et le routage : **(1.5point)**

vysh

configure terminal

interface eth0

ip address 10.1.1.1/24

interface eth1

ip address 10.1.2.1/24

exit

ip route 10.100.1.0 255.255.255.0 10.1.1.2

ip route 10.100.2.0 255.255.255.0 10.1.2.2

exit

exit

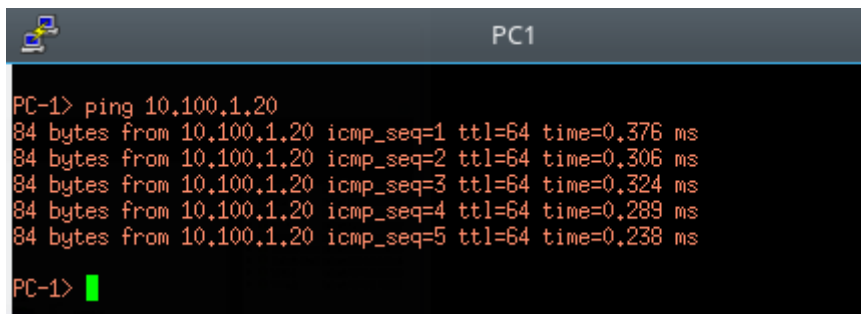
copy run start

exit

backup

- Valider le fonctionnement du réseau à l'aide de commandes de diagnostic (ping) : **(0.5point)**

pc1 → pc2



pc1 → pc3

```
PC-1> ping 10.100.2.10
84 bytes from 10.100.2.10 icmp_seq=1 ttl=61 time=2.109 ms
84 bytes from 10.100.2.10 icmp_seq=2 ttl=61 time=2.108 ms
84 bytes from 10.100.2.10 icmp_seq=3 ttl=61 time=1.982 ms
84 bytes from 10.100.2.10 icmp_seq=4 ttl=61 time=1.596 ms
84 bytes from 10.100.2.10 icmp_seq=5 ttl=61 time=3.103 ms
```

pc1 → pc4

```
PC-1> ping 10.100.2.20
84 bytes from 10.100.2.20 icmp_seq=1 ttl=61 time=1.841 ms
84 bytes from 10.100.2.20 icmp_seq=2 ttl=61 time=1.524 ms
84 bytes from 10.100.2.20 icmp_seq=3 ttl=61 time=3.454 ms
84 bytes from 10.100.2.20 icmp_seq=4 ttl=61 time=2.465 ms
84 bytes from 10.100.2.20 icmp_seq=5 ttl=61 time=1.844 ms
```

pc1 → routeur

```
PC-1> ping 10.1.1.1
84 bytes from 10.1.1.1 icmp_seq=1 ttl=63 time=1.156 ms
84 bytes from 10.1.1.1 icmp_seq=2 ttl=63 time=1.007 ms
84 bytes from 10.1.1.1 icmp_seq=3 ttl=63 time=1.256 ms
84 bytes from 10.1.1.1 icmp_seq=4 ttl=63 time=0.876 ms
84 bytes from 10.1.1.1 icmp_seq=5 ttl=63 time=1.006 ms

PC-1> ping 10.1.2.1
84 bytes from 10.1.2.1 icmp_seq=1 ttl=63 time=1.208 ms
84 bytes from 10.1.2.1 icmp_seq=2 ttl=63 time=3.095 ms
84 bytes from 10.1.2.1 icmp_seq=3 ttl=63 time=0.899 ms
84 bytes from 10.1.2.1 icmp_seq=4 ttl=63 time=2.000 ms
84 bytes from 10.1.2.1 icmp_seq=5 ttl=63 time=1.054 ms
```

- Quel est l'intérêt des commandes : `sysctl -w net.ipv4.ip_forward=1` `sudo iptables -t nat -A POSTROUTING -o <interface_public> -j MASQUERADE` **(0.5point)**

Ces commandes permettent d'activer le routage et le forwarding de port.

b) Utiliser la commande `trace <adresse_pc>` pour noter le chemin entre le pc1 et pc3. **(0.5point)**

```
PC-1> trace 10.100.2.10
trace to 10.100.2.10, 8 hops max, press Ctrl+C to stop
 1  10.100.1.1  0.351 ms  0.308 ms  0.434 ms
 2  10.1.1.1   1.292 ms  1.541 ms  1.729 ms
 3  10.1.2.2   1.603 ms  1.202 ms  0.998 ms
 4  *10.100.2.10 1.512 ms
```

c) Configurer un tunnel GRE entre les deux sites de l'entreprise. **(2points)**

VPN1:

```
sudo ip tunnel add t0 mode gre remote 10.1.2.2 local 10.1.1.2
sudo ip link set dev t0 up
sudo ip address add 172.20.1.1/24 dev t0
sudo ip route add 10.100.2.0/24 dev t0
```

VPN2:

```
sudo ip tunnel add t1 mode gre remote 10.1.1.2 local 10.1.2.2
sudo ip link set dev t1 up
sudo ip address add 172.20.1.2/24 dev t1
sudo ip route add 10.100.1.0/24 dev t1
```

- Valider le fonctionnement du tunnel à l'aide de commandes de diagnostic. (0.25point)

VPN1:

```
gns3@box:~$ ip link show t0
9: t0@NONE: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1476 qdisc noqueue state UNKNOWN
    mode DEFAULT
    link/gre 10.1.1.2 peer 10.1.2.2
gns3@box:~$ ip route show
default via 10.1.1.1 dev eth1
10.1.1.0/24 dev eth1 proto kernel scope link src 10.1.1.2
10.100.1.0/24 dev eth0 proto kernel scope link src 10.100.1.1
10.100.2.0/24 dev t0 scope link
127.0.0.1 dev lo scope link
172.20.1.0/24 dev t0 proto kernel scope link src 172.20.1.1
gns3@box:~$ ping -c 4 172.20.1.2
PING 172.20.1.2 (172.20.1.2): 56 data bytes
64 bytes from 172.20.1.2: seq=0 ttl=64 time=1.485 ms
64 bytes from 172.20.1.2: seq=1 ttl=64 time=2.517 ms
64 bytes from 172.20.1.2: seq=2 ttl=64 time=1.097 ms
64 bytes from 172.20.1.2: seq=3 ttl=64 time=1.127 ms

--- 172.20.1.2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 1.097/1.556/2.517 ms
```

VPN2:

```
gns3@box:~$ ip link show t1
9: t1@NONE: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1476 qdisc noqueue state UNKNOWN
    mode DEFAULT
    link/gre 10.1.2.2 peer 10.1.1.2
gns3@box:~$ ip route show
default via 10.1.2.1 dev eth1
10.1.2.0/24 dev eth1 proto kernel scope link src 10.1.2.2
10.100.1.0/24 dev t1 scope link
10.100.2.0/24 dev eth0 proto kernel scope link src 10.100.2.1
127.0.0.1 dev lo scope link
172.20.1.0/24 dev t1 proto kernel scope link src 172.20.1.2
gns3@box:~$ ping -c 4 172.20.1.1
PING 172.20.1.1 (172.20.1.1): 56 data bytes
64 bytes from 172.20.1.1: seq=0 ttl=64 time=0.993 ms
64 bytes from 172.20.1.1: seq=1 ttl=64 time=2.219 ms
64 bytes from 172.20.1.1: seq=2 ttl=64 time=2.226 ms
64 bytes from 172.20.1.1: seq=3 ttl=64 time=1.420 ms

--- 172.20.1.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.993/1.714/2.226 ms
gns3@box:~$
```

- Utiliser la commande **trace <adresse_pc>** pour noter le chemin entre le pc1 et pc3. Que remarquez-vous ? (0.25point)

```
PC-1> trace 10.100.2.10
trace to 10.100.2.10, 8 hops max, press Ctrl+C to stop
 1  10.100.1.1  1.204 ms  0.889 ms  0.538 ms
 2      * * *
 3  *10.100.2.10  4.085 ms
```

Le chemin du paquet n'est plus visible du à l'utilisation du tunnel.

d) Quelle est la valeur de la MTU dans le tunnel ? Expliquer le résultat. **(0.5point)**

```
gns3@box:~$ ip address show eth1
6: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP ql
en 1000
    link/ether 0c:77:97:6b:40:01 brd ff:ff:ff:ff:ff:ff
    inet 10.1.1.2/24 scope global eth1
        valid_lft forever preferred_lft forever
gns3@box:~$ ip address show t0
9: t0@NONE: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1476 qdisc noqueue state UNKNOWN

    link/gre 10.1.1.2 peer 10.1.2.2
    inet 172.20.1.1/24 scope global t0
        valid_lft forever preferred_lft forever
```

MTU (Maximum Transmission Unit) : correspond à la taille de la plus grande unité de données de protocole de couche réseau qui peut être communiquée dans une seule transaction de réseau. Dans le cas de TCP/IP avec ETHERNET cette valeur est égale à 1500 octets (comme le montre le figure de l'affichage de l'interface eth1)

Dans le cas tu tunnel, la valeur du MTU est 1476 octets.

$1500 - 24 = 1476$ (les 24 octets correspondent à l'entête du protocole GRE qui va encapsuler la paquet ip).

e) Démarrer une capture de trafic réseau dans VPN1 avec la commande **(1point)**

sudo tcpdump -vi eth1 proto gre

Ensuite lancer un ping à partir du PC1 vers le PC3 :

ping 10.100.2.10

```
gns3@box:~$ sudo tcpdump -vi eth1 proto gre
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
13:44:51.630323 IP (tos 0x0, ttl 63, id 36961, offset 0, flags [DF], proto GRE (47), length 108)
  10.1.1.2 > 10.1.2.2: GREv0, Flags [none], length 88
    IP (tos 0x0, ttl 63, id 48469, offset 0, flags [none], proto ICMP (1), length 84)
      10.100.1.10 > 10.100.2.10: ICMP echo request, id 21949, seq 1, length 64
13:44:51.638586 IP (tos 0x0, ttl 62, id 10163, offset 0, flags [DF], proto GRE (47), length 108)
  10.1.2.2 > 10.1.1.2: GREv0, Flags [none], length 88
    IP (tos 0x0, ttl 63, id 48469, offset 0, flags [none], proto ICMP (1), length 84)
      10.100.2.10 > 10.100.1.10: ICMP echo reply, id 21949, seq 1, length 64
13:44:52.640363 IP (tos 0x0, ttl 63, id 37014, offset 0, flags [DF], proto GRE (47), length 108)
  10.1.1.2 > 10.1.2.2: GREv0, Flags [none], length 88
    IP (tos 0x0, ttl 63, id 48470, offset 0, flags [none], proto ICMP (1), length 84)
      10.100.1.10 > 10.100.2.10: ICMP echo request, id 22205, seq 2, length 64
13:44:52.642480 IP (tos 0x0, ttl 62, id 10453, offset 0, flags [DF], proto GRE (47), length 108)
  10.1.2.2 > 10.1.1.2: GREv0, Flags [none], length 88
    IP (tos 0x0, ttl 63, id 48470, offset 0, flags [none], proto ICMP (1), length 84)
      10.100.2.10 > 10.100.1.10: ICMP echo reply, id 22205, seq 2, length 64
13:44:53.643804 IP (tos 0x0, ttl 63, id 37024, offset 0, flags [DF], proto GRE (47), length 108)
  10.1.1.2 > 10.1.2.2: GREv0, Flags [none], length 88
    IP (tos 0x0, ttl 63, id 48471, offset 0, flags [none], proto ICMP (1), length 84)
      10.100.1.10 > 10.100.2.10: ICMP echo request, id 22461, seq 3, length 64
```

Commenter le résultat

La connectivité entre les deux interfaces du tunnel passe à travers le réseau réel (10.1.1.0/24 et 10.1.2.0/24)

f) Quels sont les inconvénients d'un tunnel GRE ? Dans quel cas ce type de tunnel peut être utile ? **(1point)**

Le tunnel GRE n'est pas sécurisé :

- Il ne permet de chiffrer les données qui passent dans le tunnel;
- Il ne permet pas l'authentification des extrémités du tunnel,

Partie 2 : VPN site-à-site avec OpenVPN

a) Supprimer le tunnel précédemment configuré sur la passerelle. **(2points)**

VPN1 :

sudo ip link set dev t0 down

sudo ip tunnel del t0

VPN 2:

sudo ip link set dev t1 down

sudo ip tunnel del t1

b) Démarrer d'abors SSH en tapant la commande suivante dans les passerlles VPN1 et VPN2 :

sudo /usr/local/etc/init.d/openssh start

```

gns3@box:~$ sudo /usr/local/etc/init.d/openssh start
Generating public/private rsa key pair.
Your identification has been saved in /usr/local/etc/ssh/ssh_host_rsa_key.
Your public key has been saved in /usr/local/etc/ssh/ssh_host_rsa_key.pub.
The key fingerprint is:
47:53:21:15:9e:05:2d:14:f3:41:17:20:7b:2a:81:4e root@box
The key's randomart image is:
+---[RSA 2048]---+
|                |
|      .+X0=.o|  |
|      .++=.o |  |
|    E .o.oo. |  |
|   o ...o    |  |
|  .S...      |  |
|   ++       |  |
|-----+-----|
Generating public/private dsa key pair.
Your identification has been saved in /usr/local/etc/ssh/ssh_host_dsa_key.
Your public key has been saved in /usr/local/etc/ssh/ssh_host_dsa_key.pub.
The key fingerprint is:
ff:2a:24:93:00:8c:f2:d4:7b:b1:30:f2:4d:0b:75:e9 root@box
The key's randomart image is:
+---[DSA 1024]---+
| o . . . . . |
| o = = o . . |
| . o + B =   |
| . + = E     |
|   o .S      |
|    + . .    |
|    + .      |
|    . .      |
|    . . . .  |
|-----+-----|
Generating public/private ecdsa key pair.
Your identification has been saved in /usr/local/etc/ssh/ssh_host_ecdsa_key.
Your public key has been saved in /usr/local/etc/ssh/ssh_host_ecdsa_key.pub.
The key fingerprint is:
8f:dc:a8:43:17:93:37:bd:3f:de:8f:7b:b7:8a:2c:dd root@box
The key's randomart image is:
+---[ECDSA 256]---+
|                |
|      + o .     |
|    S+ . .      |
|   . . . = .    |
|  . . + + . .   |
| . . . . .Eooo|
| . . . o o=*|
|-----+-----|
Generating public/private ed25519 key pair.

```

- c) Créer une clé partagée de chiffrement avec OpenVPN sur une des passerelles, et copier cette clé sur l'autre passerelle (via la commande SSH : scp). **(2points)**
Valider le fonctionnement du tunnel à l'aide de commandes de diagnostic (affichage des adresses des interfaces, tables de routage, connexion client-à-client du tunnel).

VPN1 :

openvpn --genkey --secret cle.key

scp cle.key user@10.1.2.2:/home/gns3

```
gns3@box:~$ openvpn --genkey --secret cle.key
gns3@box:~$ sudo scp cle.key gns3@10.1.2.2:/home/gns3
gns3@10.1.2.2's password:
cle.key                                100% 636    0.6KB/s   00:00
```

VPN2: vérifier que la cle a été copiée

```
gns3@box:~$ ls
cle.key
```

- d) Configurer un tunnel sécurisé entre les deux sites de l'entreprise à l'aide la clé partagée de chiffrement précédemment créée. **(3points)**

VPN1 :

sudo openvpn --dev tun0 --local 10.1.1.2 --remote 10.1.2.2 --ifconfig 172.20.1.1 172.20.1.2 --secret cle.key

```
gns3@box:~$ sudo openvpn --dev tun0 --local 10.1.1.2 --remote 10.1.2.2 --ifconfig
172.20.1.1 172.20.1.2 --secret cle.key
OpenVPN 2.2.2 i686-pc-linux-gnu [SSL] [LZO2] [EPOLL] [P
KCS11] [eurephia] built on Mar  9 2012
IMPORTANT: OpenVPN's default port number is now 1194, b
ased on an official port number assignment by IANA. OpenVPN 2.0-beta16 and earl
ier used 5000 as the default port.
NOTE: OpenVPN 2.1 requires '--script-security 2' or hig
her to call user-defined scripts or executables
TUN/TAP device tun0 opened
/usr/local/sbin/ip link set dev tun0 up mtu 1500
/usr/local/sbin/ip addr add dev tun0 local 172.20.1.1 p
UDIPv4 link local (bound): 10.1.1.2:1194
UDIPv4 link remote: 10.1.2.2:1194
read UDIPv4 [ECONNREFUSED]: Connection refused (code=111
Peer Connection Initiated with 10.1.2.2:1194
WARNING: 'ifconfig' is used inconsistently, local='ifco
nfig 172.20.1.1 172.20.1.2', remote='ifconfig 172.20.1.2 172.20.1.1'
Initialization Sequence Completed
```

VPN2

`sudo openvpn --dev tun1 --local 10.1.2.2 --remote 10.1.1.2 --ifconfig 172.20.1.2 172.20.1.1 --secret cle.key`

```
gns3@box:~$ sudo openvpn --dev tun1 --local 10.1.2.2 --remote 10.1.1.2 --ifconfig 172.20.1.2 172.20.1.1 --secret cle.key
OpenVPN 2.2.2 i686-pc-linux-gnu [SSL] [LZO2] [EPOLL] [PKCS11] [eurephia] built on Mar  9 2012
IMPORTANT: OpenVPN's default port number is now 1194, based on an official port number assignment by IANA. OpenVPN 2.0-beta16 and earlier used 5000 as the default port.
NOTE: OpenVPN 2.1 requires '--script-security 2' or higher scripts or executables
TUN/TAP device tun1 opened
/usr/local/sbin/ip link set dev tun1 up mtu 1500
/usr/local/sbin/ip addr add dev tun1 local 172.20.1.1 peer 172.20.1.2
UDPV4 link local (bound): 10.1.2.2:1194
UDPV4 link remote: 10.1.1.2:1194
Peer Connection Initiated with 10.1.1.2:1194
Initialization Sequence Completed
```

e) Commenter la différence entre les deux méthodes. **(1point)**

Contrairement au protocole GRE, le protocole OPENVPN, utilise des algorithmes de chiffrement (AES) pour garantir la confidentialité et l'authentification entre les deux extrémités à travers une clé publique générée.