Cisco

- SSH et Telnet
- ACL Réflexives
- Tunnel GRE (Generic routing encapsulation)
- OSPF
- VRF (Virtual Routing and Forwarding)

SSH et Telnet

Telnet

Comment activer un serveur telnet?

À l'aide de ces commandes, accéder au teminale de configuration et activer telnet, sans utilisation.

```
conf t
line vty 0 4
transport input telnet
no login
end
```

SSH

Pour l'activation on doit mettre en place plus de données que le telnet pour que ce soit une connexion sécurisée

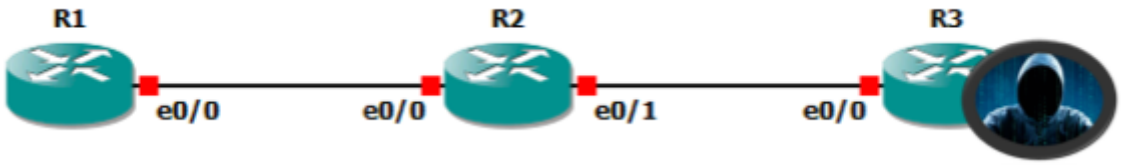
```
conf t
line vty 0 4
transport input ssh
ip domain name <VOTRE_DOMAINE>
crypto key generate rsa general modulus 2048
username toto secret toto
```

ACL Réflexives

Les Access Lists réflexives permettent de créer dynamiquement des règles permettant d'accepter les réponses uniquement aux requêtes envoyées depuis notre réseau local.

Exemple de configuration pour ce schéma

R1 étant notre réseaux local, R2 est le routeur entre le réseau local et internet et R3 étant le réseau externe, donc internet.



Sur l'interface e0/0 en IN ou sur l'interface e0/1 en OUT en mets :

```
conf t
ip access-list extended SORTANT
permit ip any any reflect MEMOIRE
exit
int e0/0
ip access-group SORTANT in
end
```

On va maintenant crée notre access list entrante.

```
conf t
ip access-list extended ENTRANT
evaluate MEMOIRE
denied ip any any
exit
int e0/1
ip access-group ENTRANT in
ènd
```

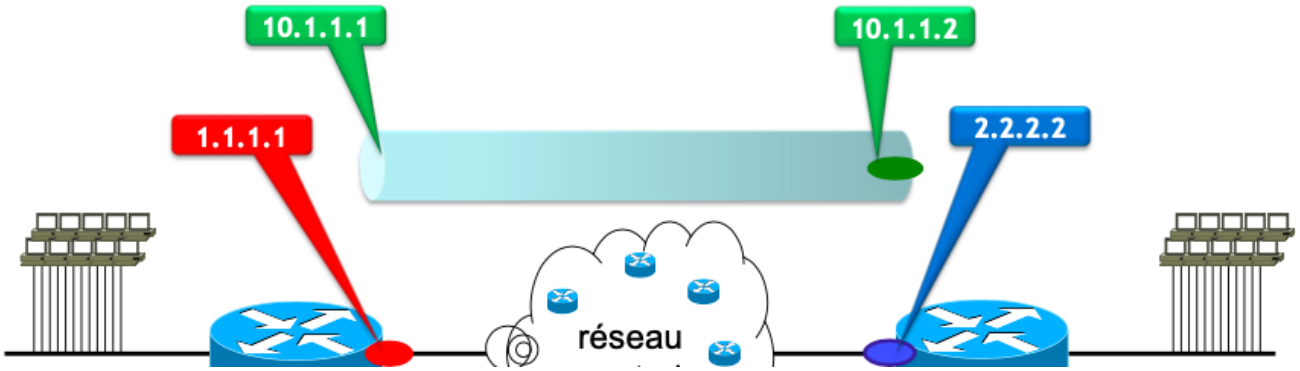
La règle ***denied ip any any*** est présente de façon implicite à la dernière ligne de chaque ACL, mais on la rajoute car elle permet de monitorer les paquets qui ont été droppés.

Chaque fois qu'un traffic passe sur e0/0 il va enrichir l'ACL mémoire, et lorqu'on aura un traffic entrant sur e0/1 il sera accepter uniquement s'il s'agit d'une réponse initié par

une requête provenant de nôtre réseau local

Tunnel GRE (Generic routing encapsulation)

L'avanatge d'un tunnel GRE permet de forcer le passage entre deux réseau à passer vers le tunnel.



Avec cet exemple précis, dans l'éventualité ou le réseau 1.1.1.1 et 2.2.2.2 peuvent déjà communiquer, voici comment rajouté le tunnel. (1.1.1.1 et 2.2.2.2 ont besoin de pouvoir communiquer le tunnel reste virtuel, et nous avons besoin d'une liaison physique)

```
Inerface tunnel 0
tunnel source 1.1.1.1
tunnel destination 2.2.2.2
ip address 10.1.1.1 255.255.255.252
```

```
Interface tunnel 0
tunnel source 2.2.2.2
tunnel destination 1.1.1.1
ip address 10.1.1.2 255.255.255.252
```

OSPF

L'ospf est un protocole de routage dynamique.

Activation sur les interfaces

Prenons l'example des interface e0/0, e0/1, e0/3.

```
conf t
int range e0/0-3
ip ospf 1 area 0
```

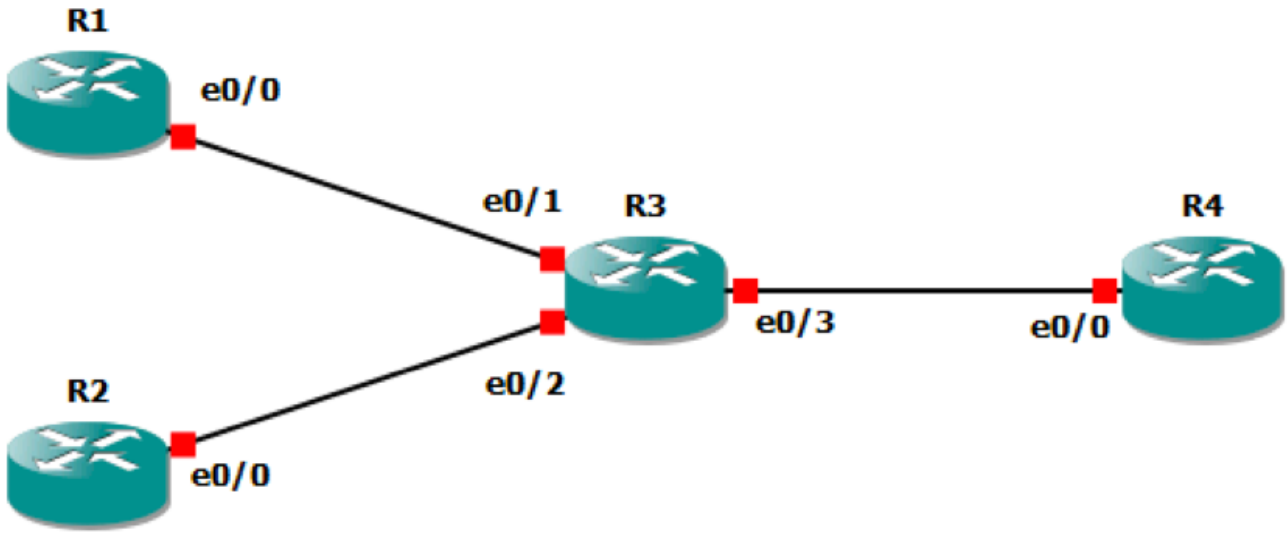
Pour vérifier notre table de routage ospf on utilise la commande :

```
show ip route ospf
```

VRF (Virtual Routing and Forwarding)

Les VRF nous permettent permettent de séraper virtuellement deux clien ayant la même ip d'atteindre un serveur. On ne veut pas que ces deux clients là se connaissent.

Sur ce shcéma imaginons cette configuration.



Sur R1

- e0/0 13.0.0.1/24
- loopback1 1.1.1.1/32

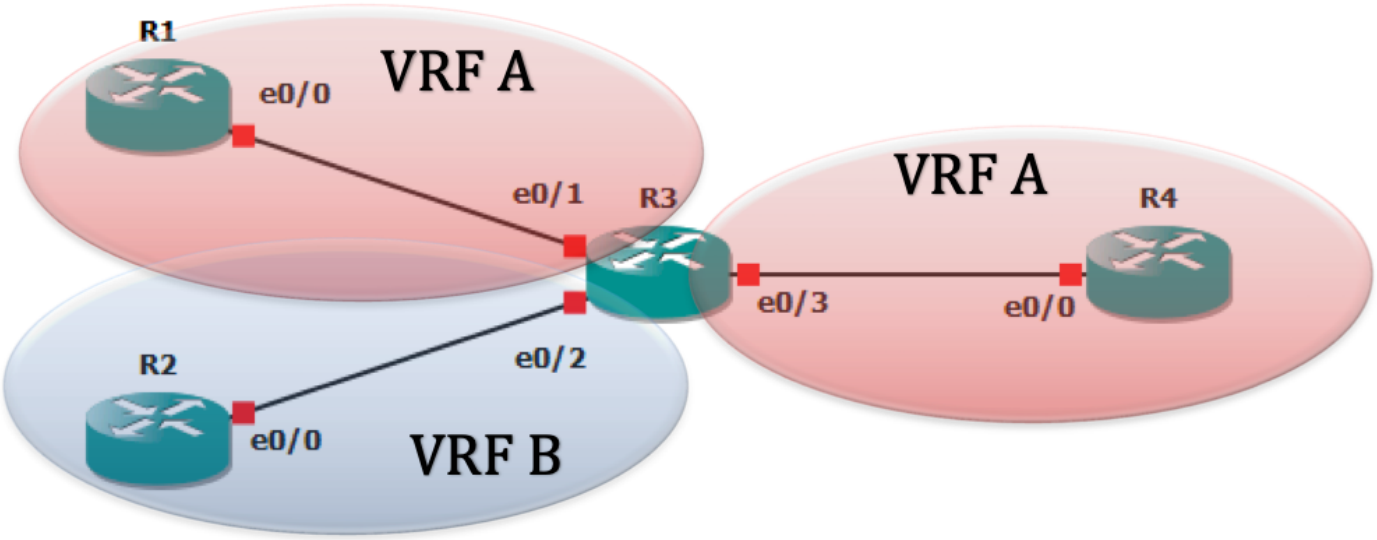
Sur R2

- e0/0 23.0.0.2/24
- loopback1 1.1.1.1/32

Sur R4

- e0/0 34.0.0.4

Nous ne mettons aucune adresses sur R3 car elle vont dispraître avec l'activation des vrf, nous allons les activer d'abord.
Imaginons la configuration suivante



Sur R3

`ip vrf ROUGE`
`rd 13:34`

Ensuite

`ip vrf BLEU`
`rd 23:23`

Vous pouvez vérifier la configuration avec `show ip vrf`

Pour affecter les interfaces au différentes vrf. En mode configuration terminale on fait :

`int range e0/1,e0/3`
`ip vrf forwarding ROUGE`

Et

`int e0/2`
`ip vrf forwarding ROUGE`

Refaire un `show ip vrf` pour voir que les interfaces ont bien été affectées.

Après cela, vous pouvez affecter les addresses ip aux interfaces de R3.

Ensuite faudra configurer les routes statiques pour que la loopback de R1 puisse accéder à la loopback de R4, et inversement, tout cela via les vrf soit :

Sur R1 en mode configuration, on fait `ip route 4.4.4.4 255.255.255.255 13.0.0.3`
Sur R4 en mode conf, on fait `ip route 1.1.1.1 255.255.255.255 34.0.0.3`

Ensuite il faudra apprendre à R3 d'aller au réseau 4.4.4.4 et aux deux réseaux 1.1.1.1 via la VRF.

Sur R3

`conf t`
`ip route vrf ROUGE 1.1.1.1 255.255.255.255 13.0.0.1`
`ip route vrf ROUGE 4.4.4.4 255.255.255.255 34.0.0.4`
`ip route vrf BLEU 1.1.1.1 255.255.255.255 23.0.0.2`

Pour vérifier les route vrf on utilise `show ip route vrf ROUGE` ou `show ip route vrf BLEU`

Pour tester on ping depuis R4 1.1.1.1 `ping 1.1.1.1 source 4.4.4.4` et on voit que le ping répond.
Si on ne précise pas la source le ping ne passe pas, parce qu'on a pas appris à R4 à joindre le 13.0.0.0, mais juste 4.4.4.4

En activant le debug icmp `debug ip icmp` sur R1 et R2 on voit bien que c'est R1 qui réponds au ping dont la source est 4.4.4.4 parce que R1 et R4 sont la même vrf.

De R3 si on fait `ping vrf ROUGE 1.1.1.1` il n'y a que R1 qui répond lorsqu'on monitor le debug icmp. Et avec `ping vrf BLEU 1.1.1.1` c'est R2 qui réponds.

Le but était qu'avec deux client R1 et R2 ayant un même réseau, simuler ici avec les loopback, puissent exister et accéder au serveur R4 sans conflits avec R1 et R4 qui communiquent et sans que R4 connaissent l'existance de R2 même si c'est la même adresse ip que R1.

Dans le cas ou il y'a besoin de faire des sous interface sur le routeur:

`int g0/0.100`
`encapsulation dot1q 100`
`ip vrf forwarding Red`
`ip address 10.0.0.1`

Sur les routeur externes

`conf t`
`router ospf 1`
`network 0.0.0.0 0.0.0.0 area 0`
`end`

Sur les routeurs vrf

`conf t` `router ospf 1 vrf ROUGE` `router ospf 2 vrf BLEU` `int range e0/1, e0/0.100` `ip ospf 1 area 0` `int e0/2, e0/0.200` `ip ospf 2 area 0` `end`