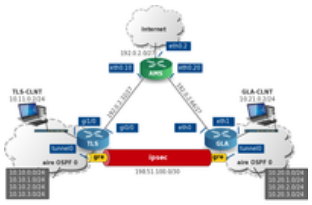
Routage OSPF & VPN IPsec site à site

Philippe Latu
philippe.latu(at)inetdoc.net

http://www.inetdoc.net

Résumé



La mise en œuvre d'un réseau privé virtuel (VPN) avec le jeu de protocoles IPsec est un exercice qui peut rapidement tourner au calvaire compte tenu du nombre des combinaisons possibles entre les différents algorithmes et autres paramètres. C'est particulièrement vrai si on utilise des systèmes d'interconnexion différents. Cet article est une synthèse rédigée après avoir «survécu» à la configuration d'un VPN IPsec avec routage OSPF entre deux sites distants ; le premier utilisant un routeur Cisco™ et le second utilisant un routeur GNU/Linux.

Après avoir présenté le contexte et rappelé sommairement les caractéristiques du jeu de protocoles IPsec, on précise les choix de combinaisons de protocoles puis on décrit étape par étape le processus de déploiement de la topologie type d'intercconnexion réseau.

Table des matières

1. Copyright et Licence

1.1. Méta-information

Cet article est écrit avec DocBook[1] XML sur un système Debian GNU/Linux[2]. Il est disponible en version imprimable au format PDF : `site2site-ipsecvpn.pdf`[3].

1.2. Conventions typographiques

Tous les exemples d'exécution des commandes sont précédés d'une invite utilisateur ou prompt spécifique au niveau des droits utilisateurs nécessaires sur le système.

- Toute commande précédée de l'invite `$` ne nécessite aucun privilège particulier et peut être utilisée au niveau utilisateur simple.
- Toute commande précédée de l'invite `#` nécessite les privilèges du super-utilisateur.

[1] http://www.docbook.org
[2] http://www.debian.org
[3] http://www.inetdoc.net/pdf/site2site-ipsecvpn.pdf

2. IPsec, les principes

Aborder la thématique des réseaux privés virtuels (Virtual Private Networks ou VPNs) utilisant IPsec sans préciser les conditions dans lesquelles ces techniques sont utilisées est une gageure. Pire encore, justifier les choix effectués sur l'architecture réseau sans une présentation succincte du jeu de protocoles et d'algorithmes de cryptographie mis en œuvre rendrait la suite du document inexploitable. C'est la raison pour laquelle on présente ci-après un minimum d'éléments sur IPsec.

Il existe déjà un grand nombre de documentations en ligne d'excellente qualité sur le sujet. Alors, pourquoi un énième rappel sur les protocoles IPsec ? La raison d'être de cette section est de servir de référence sur les choix effectués par la suite. Lors des recherches en ligne sur IPsec, on trouve trop souvent des copies de fichiers de configuration peu ou pas commentés qui répondent à un contexte singulier. Ici, l'objectif est justifier les options retenues en fonction du contexte défini.

Le retour de la triade CIA ... ou presque

Au niveau le plus général, en sécurité des systèmes d'information (Information security[4]), la lettre **C** correspond à Confidentiality, la lettre **I** correspond à Integrity et la lettre **A** correspond à Availability. Dans le contexte de ce document, seule l'interprétation de la lettre **A** diffère ; le terme Availability est remplacé par Authentication. La «triade CIA» est donc aussi un moyen mnémotechnique pertinent dans le contexte présent. Voir l'article Concepts élémentaires en sécurité de l'information[5].

IPsec n'est pas un protocole unique mais un ensemble de protocoles définis par un organisme très connu : The Internet Engineering Task Force (IETF)[6]. Le but est de fournir un haut niveau de sécurité dans l'échange des paquets IP. On se situe donc au niveau de la couche réseau de la modélisation. Les services doivent utiliser la cryptographie et être interopérables. Les points clés sont :

- Confidentialité : l'émetteur chiffre les paquets avant de les transmettre sur l'Internet.
- Intégrité : le récepteur vérifie que les paquets reçus n'ont pas été altérés lors de la transmission.
- Authentification : le récepteur vérifie l'identité de l'émetteur. Plusieurs méthodes d'authentification sont disponibles.
  - Clés partagées ou preshared keys. La «clé secrète» est déposée à l'avance sur chaque extrémité. La même clé est utilisée durant les phases d'établissement des associations de sécurité.
  - Certificats numériques signés ou signed digital certificates. Chaque extrémité détient un certificat propre contenant un couple clé privée/clé publique. Une extrémité utilise la clé privée pour signer les données à émettre tandis que l'autre extrémité utilise la clé publique pour vérifier les données reçues.
  - Authentification externe. On fait appel à un service externe tel que Keberos ou Radius qui permet d'accéder aux différentes formes de couple nom d'utilisateur/mot de passe : One Time Password (OTP), token, biométrie, etc.

  > **Note**
  >
  > Dans le contexte de ce document, la méthode d'authentification utilise les clés partagées. Le trafic doit être sécurisé de façon permanente et continue entre deux sites «fixes».

- Anti rejeu : le récepteur examine les paquets et rejette les paquets périmés (reçus en dehors de la fenêtre de temps prévue) et/ou répétés.

Les conditions d'utilisation des points ci-dessus sont négociées à l'aide du protocole IKE (Internet Key Exchange[7]) et de la mise en place d'associations de sécurité SA (Security association[8]). Il est aussi

[4] http://en.wikipedia.org/wiki/Information_security
[5] http://www.inetdoc.net/articles/infosecconcepts/
[6] http://www.ietf.org/
[7] http://en.wikipedia.org/wiki/Internet_Key_Exchange
[8] http://en.wikipedia.org/wiki/Security_association

possible de configurer manuellement les politiques de chiffrement et les algorithmes. Dans ce dernier cas, le protocole IKE est inutile.

L'échange de clés Diffie-Hellman

Cette méthode est utilisée dans les échanges du protocole IKE pour générer un secret partagé entre les deux extrémités de façon sécurisée. Avec cette méthode, un «observateur» ayant capturé les échanges IKE ne peut pas remonter au secret partagé.

Le secret partagé est utilisé pour calculer une valeur qui sert ensuite aux calculs des clés utilisées dans les phases 1 et 2 du protocole IKE. Des groupes auxquels ont été associé une longueur de clé et une fonction de chiffrement doivent être choisis pour mettre en œuvre l'échange de clés. Plus la clé comprend un nombre de bits important, plus le secret est «solide». En contre partie, le temps de calcul est lui aussi plus important. Cette méthode est appelée à chaque établissement ou renouvellement d'association de sécurité.

Il faut ajouter à ce mécanisme d'échange de clés la technique appelée Perfect Forward Secrecy (PFS). Avec cette technique, les calculs de clés sont plus sûrs dans la mesure où les nouveaux calculs de renouvellement de SA ne se font pas à partir des anciennes clés.

Les deux protocoles de sécurisation IPsec : AH & ESP

En complément du protocole IKE qui établit un tunnel, IPsec s'appuie sur deux services qui peuvent être utilisés seuls ou associés.

Authentication Header, AH, protocole n°51
: Le protocole AH authentifie l'émetteur des données, contrôle l'intégrité du paquet IP (en-tête et charge utile) et assure le service anti rejeu. Il couvre donc les lettres **I** et **A** de la triade CIA.

  Les traitements associés utilisent des algorithmes de «hachage» tels que Message Digest 5 (MD5) ou Secure Hash Algorithm (SHA-1, SHA-256, etc.). Un algorithme de hachage est assocé à une clé issue de la méthode d'authentification choisie pour constituer un Hash base Message Authentication Code (HMAC).

  L'en-tête AH est inséré à la suite de l'en-tête IP pour garantir l'intégrité et l'authenticité des données. On se protège ainsi contre toute altération du paquet lors de son transit.

  Des représentations graphiques détaillées de l'encapsulation du protocole AH sont fournies dans le document An Illustrated Guide to IPsec[9] pour les deux modes de fonctionnement transport et tunnel présentés ci-après.

Encapsulating Security Payload, ESP, protocole n°50
: Le protocole ESP se distingue du précédent par le chiffrement de la partie données avant encapsulation dans le paquet IP. On intègre ainsi la partie confidentialité aux fonctions de sécurisation. Ce protocole couvre donc les trois lettres de la triade CIA sur la partie charge utile du paquet.

  Les traitements associés, en plus des algorithmes de hachage cités précédemment, font intervenir des algorithmes de chiffrement tels que Triple Data Encryption Standard (3DES) ou Advanced Encryption Standard (AES).

  Comme dans le cas du protocole AH, des représentations graphiques détaillées de l'encapsulation sont fournies dans le document An Illustrated Guide to IPsec[10] pour les deux modes de fonctionnement transport et tunnel présentés ci-après.

Les deux modes de fonctionnement IPsec : Transport & Tunnel

host-to-host transport mode, mode transport
: Dans ce mode, les échanges de paquets IP sont sécurisés entre deux extrémités. Seule la charge utile ou payload est concernée par les traitements et l'en-tête du paquet IP est préservé pour permettre au routage de fonctionner de façon transparente.

[9] http://www.unixwiz.net/techtips/iguide-ipsec.html
[10] http://www.unixwiz.net/techtips/iguide-ipsec.html

network-to-network tunnel mode, mode tunnel
: Dans ce mode, les échanges de paquets IP sont sécurisés de réseau à réseau. La totalité du paquet IP (en-tête + charge utile) est encapsulée et un nouvel en-tête de paquet IP est créé.

Le choix entre ces deux modes de fonctionnement est un point essentiel dans la mise en œuvre d'un VPN IPsec. L'argumentation tourne souvent autour de l'inclusion ou non de l'en-tête de paquet IP dans les fonctions de sécurité de la triade CIA et du nombre de réseaux à faire transiter via le VPN.

L'établissement d'une association de sécurité : IKE & ISAKMP

L'objectif du mécanisme d'échange IKE, via le protocole ISAKMP, est de permettre à deux extrémités en communication d'établir une association de sécurité. Cette association demande au moins deux étapes.

IKE phase 1, ISAKMP-SA
: Durant cette première phase, il y a établissement d'une session sécurisée ISAKMP entre les deux extrémités que l'on appelle ISAKMP SA. C'est dans cette même phase que sont configurés les jeux de traitements utilisables (transform-set) : algorithmes de hachage, algorithmes de chiffrement et méthode d'authentification.

IKE phase 1.5
: Cette «phase intermédiaire» ne peut se dérouler que sous la protection du tunnel ISAKMP établi en phase 1. Elle est aussi appelée Extended Authentication (XAUTH). Ici, l'authentification utilisateur est étendue pour fournir des éléments de configuration supplémentaires à un client VPN : adresses IPv4 ou IPv6, masque réseau, serveur DNS, etc.

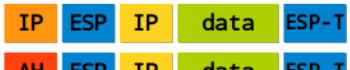
Note

Dans le contexte de ce document, cette phase est sans objet dans la mesure où les deux extrémités du tunnel ISAKMP sont des routeurs dont les paramètres de configuration sont déjà connus.

IKE phase 2, IPsec-SA
: Comme dans le cas précédent, cette phase ne peut se dérouler que sous la protection du tunnel ISAKMP établi en phase 1. À la différence de la phase 1, on ne parle plus d'une association de sécurité bidirectionnelle utilisant la même clé partagée entre les deux extrémités. Ici, il faut établir deux associations unidirectionnelles entre les mêmes extrémités. Une fois que ces deux SA sont en place, le tunnel IPsec est actif. Les paramètres négociés sont quasiment les mêmes que lors de la phase 1 : algorithme de hachage, algorithme de chiffrement, groupe Diffie Hellman, informations anti rejeu, durée de vie de l'association de sécurité.

La synthèse sur l'encapsulation

Tableau 1. Encapsulation : modes de fonctionnement & protocoles

Protocole	Mode Transport	Mode Tunnel
AH		
ESP		
AH+ESP		

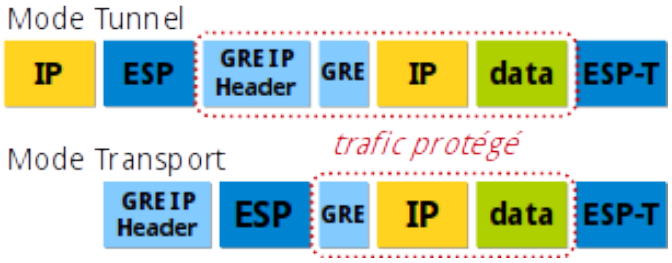
Le trafic unicast ou le tunnel GRE

Pour achever cette partie sur les principes du jeu de protocoles IPsec, il reste un dernier point très important à ajouter : IPsec seul ne peut véhiculer que le trafic unicast. C'est un critère essentiel à prendre en compte lors du choix d'une solution d'encapsulation.

Dans le contexte de ce document, nous sommes apparemment face à un sérieux problème sachant que tous les échanges entre instances de protocoles de routage dynamique se font en multicast. Fort heureusement, il existe une solution : le tunnel GRE.

À priori, les tunnels GRE (Generic Routing Encapsulation) n'ont aucun rapport avec le jeu de protocoles IPsec. Ce protocole, initialement développé par Cisco™, a été conçu pour encapsuler n'importe quel protocole dans un tunel IP. Il a ensuite été étendu à l'encapsulation IP dans IP. Associé à IPsec, il permet de dépasser la limitation de ce dernier au trafic unicast.

Le couple GRE + IPsec peut être assimilé à un mode tunnel sans restriction sur la nature du trafic véhiculé.



Dans la représentation ci-dessus, le champ GRE IP Header correspond à l'en-tête IP introduit par l'utilisation du tunnel GRE.

Cette même représentation montre que l'utilisation du mode transport avec le protocole ESP offre généralement un niveau de sécurité suffisant sachant que seul l'en-tête IP lié à l'utilisation du tunnel GRE est exposé. L'utilisation du mode tunnel implique l'encapsulation d'un en-tête IP supplémentaire qui pénalise l'espace réservé à la charge utile.

3. IPsec, la topologie étudiée

Le choix : mode transport ou mode tunnel

Dans la section précédente, deux points importants ont déjà été énoncés.

- IPsec ne véhicule que le trafic unicast.
- Le mode tunnel assure la communication d'un réseau à un autre. Notez bien : réseau au singulier !

L'utilisation de réseaux multiples en mode tunnel seul revient à constituer une topologie entièrement maillée avec un tunnel de communication entre chaque paire de réseau. En suivant la règle classique, pour **n** réseaux à interconnecter, il faut créer **n.(n - 1)/2** tunnels. Face au surcoût de configuration et d'administration d'un réseau maillé, il est nettement préférable d'utiliser une topologie de type Hub & Spoke pour laquelle le nombre de tunnels à créer est **n - 1**.

Il est aussi possible d'utiliser un protocole de routage dynamique entre les extrémités IPsec de façon a échanger plus librement les routes vers les différents réseaux. C'est justement cette dernière solution qui est étudiée dans ce document. Le recours au tunnel GRE s'impose donc.

> **VTI : Virtual Tunnel Interface**
>
> Sur les équipements Cisco™, la notion d'interface virtuelle de tunnel a été introduite. Dans le contexte de ce document, ce type d'interface permet «d'économiser» l'encapsulation GRE sachant que le trafic multicast est directement traité. On gagne ainsi 24 octets de charge utile par paquet transmis. Le logiciel iproute2 doit proposer un fonctionnement équivalent sur les systèmes GNU/Linux. Malheureusement, le paquet Debian correspondant ne dispose pas encore de cette fonction au moment de la rédaction de ce document.

La liste ci-dessous reprend les arguments en faveur de l'emploi du tunnel GRE.

- GRE est assimilable à un tunnel IPsec dans le sens ou le paquet IP à acheminer est entièrement encapsulé.
- GRE est sans état (stateless) et ne dispose pas de mécanisme de contrôle de flux. Ces fonctions sont assurées par les protocoles de routage ou les protocoles de couches supérieures. Avec IPsec, les phases d'échanges de clés assurent un contrôle d'état en établissant les associations de sécurité.
- GRE ajoute un en-tête minimum de 24 octets en incluant les 20 octets du nouvel en-tête IP de tunnel.

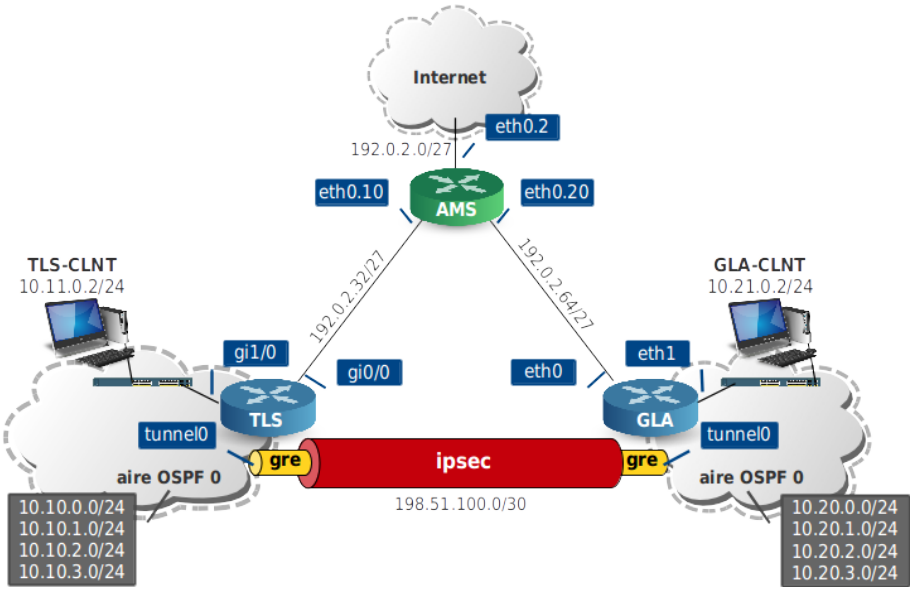
- GRE peut acheminer n'importe quel protocole de couche réseau du modèle OSI.
- GRE permet l'acheminement du trafic multicast et donc des protocoles de routage dynamique à travers le tunnel.

Pour terminer sur le choix de la solution de mise en œuvre du VPN IPsec site à site, la configuration d'un tunnel GRE puis des communications IPsec en mode transport avec le protocole ESP se fait de façon totalement symétrique entre un routeur Cisco™ et un système GNU/Linux. Si les outils et la syntaxe diffèrent entre les deux types de systèmes, les fonctionnalités sont strictement identiques.

La topologie type étudiée dans ce document reprend étape par étape l'installation de ces fonctions.

La topologie type étudiée

De façon très académique (et donc pas très originale), la topologie de base utilisée dans ce document est le triangle. Ainsi les deux extrémités des tunnels GRE et IPsec appartiennent à des réseaux différents. Du point de vue du routage dynamique, ces deux mêmes extrémités ne peuvent être adjacentes. Seul l'établissement d'un tunnel permet de véhiculer les échanges propres aux démons de routage dynamique.



Dans la topologie présentée ci-dessus, les trois routeurs ont des rôles et/ou des systèmes différents.

- Le routeur TLS est un routeur Cisco™ sur lequel on trouve une instance de routage OSPF et une extrémité de tunnel GRE + IPsec. Deux systèmes ont été étudiés : un «faux» routeur 7200 émulé via dynamips et un «vrai» routeur ASR1001.
- Le routeur AMS est un système Debian GNU/Linux qui ne fait que du routage statique et de la traduction d'adresses sources pour le trafic sortant vers l'Internet. Deux systèmes ont été étudiés : une instance de machine virtuelle sur KVM + openvswitch et un serveur HP™ DL-380.
- Le routeur GLA est aussi un système Debian GNU/Linux sur lequel on trouve une instance de routage OSPF et une extrémité de tunnel GRE + IPsec. Deux systèmes ont été étudiés : une instance de machine virtuelle sur KVM + openvswitch et un serveur HP™ DL-380.

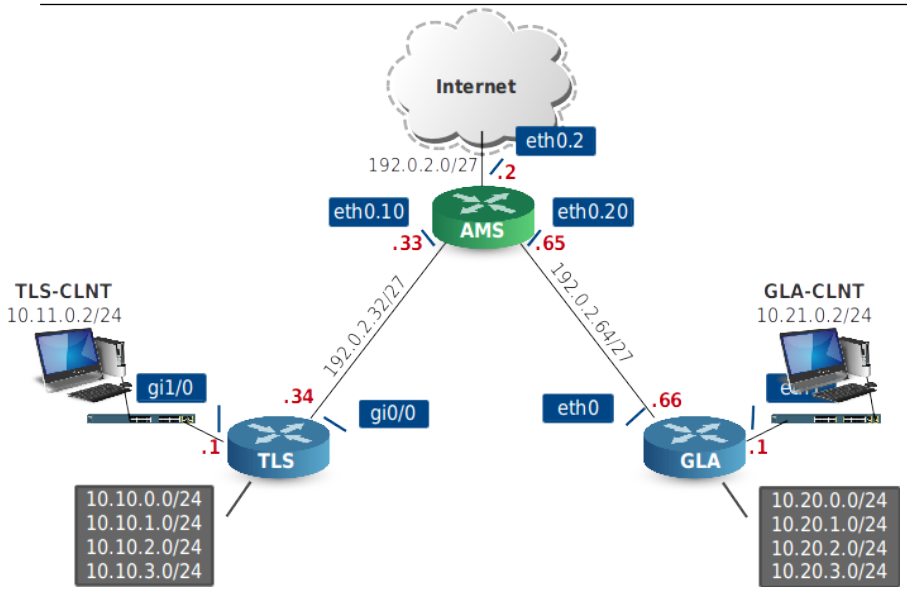
Les étapes de configuration sont décrites dans les sections suivantes.

- Configuration de la topologie de base et validation des communications au niveau réseau entre les routeurs.
- Configuration du tunnel GRE entre les deux routeurs d'extrémité.
- Configuration du protocole de routage dynamique OSPF entre les deux mêmes routeurs d'extrémité. Validation des communications entre les réseaux échangés via OSPF.
- Configuration IPsec en mode transport avec le protocole ESP ; à nouveau entre les deux mêmes routeurs d'extrémité. Validation des phases d'échanges de clé IKE et des associations de sécurité (SA).

4. La topologie de base : le triangle

Dans cette section on configure les interfaces réseau, les routes statiques et on valide les communications point à point entre les différents routeurs de la topologie triangle de base.

> **Note de l'auteur**
>
> Le contenu de cette section peut paraître totalement superflu à toute personne ayant un peu d'expérience dans le domaine. Cependant, j'ai pu constater à de très nombreuses reprises que les (étudiants|débutants) se lancent systématiquement dans la configuration des services sans avoir simplement validé les communications aux niveaux inférieurs. C'est une figure classique qui se termine immanquablement par une formule du style «Ouais j'comprend rien ; çà marche pâ (Linux|IOS) c'est de la ...». C'est donc pour lutter contre ce genre de stigmate que je m'acharne à valider les communications point à point avant d'attaquer le vif du sujet.



Le routeur AMS

- État des interfaces réseau. Le fichier de configuration est donné en annexe : Routeur `AMS`.

```
$ ip addr ls scope global
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether ba:ad:00:ca:fe:02 brd ff:ff:ff:ff:ff:ff
3: eth0.2@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether ba:ad:00:ca:fe:02 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.2/27 brd 192.0.2.31 scope global eth0.2
4: eth0.10@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether ba:ad:00:ca:fe:02 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.33/27 brd 192.0.2.63 scope global eth0.10
5: eth0.20@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether ba:ad:00:ca:fe:02 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.65/27 brd 192.0.2.95 scope global eth0.20
```

- Table de routage

```
$ ip route ls
default via 192.0.2.1 dev eth0.2
10.10.0.0/22 via 192.0.2.34 dev eth0.10
10.11.0.0/24 via 192.0.2.34 dev eth0.10
10.20.0.0/22 via 192.0.2.66 dev eth0.20
10.21.0.0/24 via 192.0.2.66 dev eth0.20
192.0.2.0/27 dev eth0.2  proto kernel  scope link  src 192.0.2.2
192.0.2.32/27 dev eth0.10  proto kernel  scope link  src 192.0.2.33
192.0.2.64/27 dev eth0.20  proto kernel  scope link  src 192.0.2.65
```

Dans la copie d'écran ci-dessus, on trouve quatre entrées de routage statique vers les réseaux `10.x.x.x`. Ces routes sont nécessaires pour que les deux autres routeurs puissent communiquer entre eux. Ces mêmes entrées permettent aussi aux autres systèmes d'accéder à Internet.

- Tests ICMP point à point sur les trois réseaux connectés au routeur

```
$ ping -c2 192.0.2.1
PING 192.0.2.1 (192.0.2.1) 56(84) bytes of data.
64 bytes from 192.0.2.1: icmp_req=1 ttl=64 time=0.786 ms
64 bytes from 192.0.2.1: icmp_req=2 ttl=64 time=0.616 ms

--- 192.0.2.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.616/0.701/0.786/0.085 ms
```

```
$ ping -c2 192.0.2.34
PING 192.0.2.34 (192.0.2.34) 56(84) bytes of data.
64 bytes from 192.0.2.34: icmp_req=1 ttl=255 time=5.14 ms
64 bytes from 192.0.2.34: icmp_req=2 ttl=255 time=4.81 ms

--- 192.0.2.34 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 4.817/4.978/5.140/0.176 ms
```

```
$ ping -c2 192.0.2.66
PING 192.0.2.66 (192.0.2.66) 56(84) bytes of data.
64 bytes from 192.0.2.66: icmp_req=1 ttl=64 time=1.53 ms
64 bytes from 192.0.2.66: icmp_req=2 ttl=64 time=1.30 ms

--- 192.0.2.66 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.305/1.417/1.530/0.118 ms
```

- Traduction d'adresses sources du trafic sortant vers l'Internet

```
# iptables -t nat -vL  POSTROUTING
Chain POSTROUTING (policy ACCEPT 244 packets, 55952 bytes)
 pkts bytes target      prot opt in     out     source               destination
 3937  241K MASQUERADE  all  --  any    eth0.2  anywhere             anywhere
```

Le routeur TLS

- État des interfaces réseau. L'extrait du fichier de configuration est donné en annexe : Routeur `TLS`.

```
TLS#sh ip int brief
Interface                  IP-Address      OK? Method Status                Protocol
Ethernet0/0                unassigned      YES NVRAM  administratively down down
GigabitEthernet0/0         192.0.2.34      YES NVRAM  up                    up
GigabitEthernet1/0         10.11.0.1       YES NVRAM  up                    up
Loopback0                  10.10.0.1       YES NVRAM  up                    up
Loopback1                  10.10.1.1       YES NVRAM  up                    up
Loopback2                  10.10.2.1       YES NVRAM  up                    up
Loopback3                  10.10.3.1       YES NVRAM  up                    up
```

- Table de routage

```
TLS#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       + - replicated route, % - next hop override

Gateway of last resort is 192.0.2.33 to network 0.0.0.0

S*    0.0.0.0/0 [1/0] via 192.0.2.33
      10.0.0.0/8 is variably subnetted, 15 subnets, 2 masks
C        10.10.0.0/24 is directly connected, Loopback0
L        10.10.0.1/32 is directly connected, Loopback0
C        10.10.1.0/24 is directly connected, Loopback1
L        10.10.1.1/32 is directly connected, Loopback1
C        10.10.2.0/24 is directly connected, Loopback2
L        10.10.2.1/32 is directly connected, Loopback2
C        10.10.3.0/24 is directly connected, Loopback3
L        10.10.3.1/32 is directly connected, Loopback3
C        10.11.0.0/24 is directly connected, GigabitEthernet1/0
L        10.11.0.1/32 is directly connected, GigabitEthernet1/0
      192.0.2.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.0.2.32/27 is directly connected, GigabitEthernet0/0
L        192.0.2.34/32 is directly connected, GigabitEthernet0/0
```

- Tests ICMP point à point sur les réseaux connectés au routeur

```
TLS#ping 192.0.2.33
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.0.2.33, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/8/12 ms
```

```
TLS#ping 10.11.0.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.11.0.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/10/20 ms
```

```
TLS#ping 10.10.0.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.0.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/12 ms
```

```
TLS#ping 10.10.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/4/12 ms
```

```
TLS#ping 10.10.2.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.2.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

```
TLS#ping 10.10.3.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.3.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/12 ms
```

- Tests ICMP vers le routeur GLA

```
TLS#ping 192.0.2.66
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.0.2.66, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/11/20 ms
```

L'adresse IP destination `192.0.2.66` correspond à l'extrémité du tunnel à établir pour constituer la liaison VPN.

Le routeur GLA

- État des interfaces réseau. Le fichier de configuration est donné en annexe : Routeur GLA.

```
$ ip addr ls scope global
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether ba:ad:00:ca:fe:20 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.66/27 brd 192.0.2.95 scope global eth0
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether ba:ad:00:ca:fe:21 brd ff:ff:ff:ff:ff:ff
    inet 10.21.0.1/24 brd 10.21.0.255 scope global eth1
4: dummy0: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
    link/ether da:32:1c:fa:f1:9c brd ff:ff:ff:ff:ff:ff
    inet 10.20.0.1/24 brd 10.20.0.255 scope global dummy0
5: dummy1: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
    link/ether 7a:ff:7c:26:ee:2b brd ff:ff:ff:ff:ff:ff
    inet 10.20.1.1/24 brd 10.20.1.255 scope global dummy1
6: dummy2: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
    link/ether 3a:dd:fd:c6:eb:7a brd ff:ff:ff:ff:ff:ff
    inet 10.20.2.1/24 brd 10.20.2.255 scope global dummy2
7: dummy3: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
    link/ether 06:23:20:ae:9f:ff brd ff:ff:ff:ff:ff:ff
    inet 10.20.3.1/24 brd 10.20.3.255 scope global dummy3
```

- Table de routage

```
$ ip route ls
default via 192.0.2.65 dev eth0
10.20.0.0/24 dev dummy0  proto kernel  scope link  src 10.20.0.1
10.20.1.0/24 dev dummy1  proto kernel  scope link  src 10.20.1.1
10.20.2.0/24 dev dummy2  proto kernel  scope link  src 10.20.2.1
10.20.3.0/24 dev dummy3  proto kernel  scope link  src 10.20.3.1
10.21.0.0/24 dev eth1  proto kernel  scope link  src 10.21.0.1
192.0.2.64/27 dev eth0  proto kernel  scope link  src 192.0.2.66
```

- Tests ICMP point à point sur les réseaux connectés au routeur

```
$ ping -c2 192.0.2.65
PING 192.0.2.65 (192.0.2.65) 56(84) bytes of data.
64 bytes from 192.0.2.65: icmp_req=1 ttl=64 time=1.10 ms
64 bytes from 192.0.2.65: icmp_req=2 ttl=64 time=1.27 ms

--- 192.0.2.65 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.106/1.190/1.275/0.091 ms
```

```
$ ping -c2 10.21.0.2
PING 10.21.0.2 (10.21.0.2) 56(84) bytes of data.
64 bytes from 10.21.0.2: icmp_req=1 ttl=64 time=1.32 ms
64 bytes from 10.21.0.2: icmp_req=2 ttl=64 time=1.24 ms

--- 10.21.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.247/1.284/1.322/0.051 ms
```

```
$ ping -c2 10.20.0.1
PING 10.20.0.1 (10.20.0.1) 56(84) bytes of data.
64 bytes from 10.20.0.1: icmp_req=1 ttl=64 time=0.072 ms
64 bytes from 10.20.0.1: icmp_req=2 ttl=64 time=0.089 ms

--- 10.20.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.072/0.080/0.089/0.012 ms
```

```
$ ping -c2 10.20.1.1
PING 10.20.1.1 (10.20.1.1) 56(84) bytes of data.
64 bytes from 10.20.1.1: icmp_req=1 ttl=64 time=0.075 ms
64 bytes from 10.20.1.1: icmp_req=2 ttl=64 time=0.083 ms

--- 10.20.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.075/0.079/0.083/0.004 ms
```

```
$ ping -c2 10.20.2.1
PING 10.20.2.1 (10.20.2.1) 56(84) bytes of data.
64 bytes from 10.20.2.1: icmp_req=1 ttl=64 time=0.089 ms
64 bytes from 10.20.2.1: icmp_req=2 ttl=64 time=0.086 ms

--- 10.20.2.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.086/0.087/0.089/0.009 ms
```

```
$ ping -c2 10.20.3.1
PING 10.20.3.1 (10.20.3.1) 56(84) bytes of data.
64 bytes from 10.20.3.1: icmp_req=1 ttl=64 time=0.083 ms
64 bytes from 10.20.3.1: icmp_req=2 ttl=64 time=0.137 ms

--- 10.20.3.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.083/0.110/0.137/0.027 ms
```

- Tests ICMP vers le routeur TLS

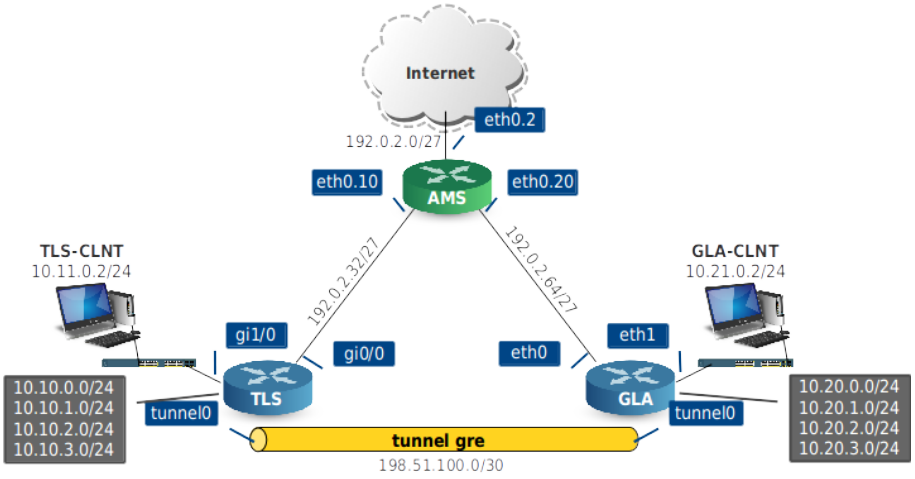
```
$ ping -c2 192.0.2.34
PING 192.0.2.34 (192.0.2.34) 56(84) bytes of data.
64 bytes from 192.0.2.34: icmp_req=1 ttl=254 time=14.6 ms
64 bytes from 192.0.2.34: icmp_req=2 ttl=254 time=14.1 ms

--- 192.0.2.34 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 14.188/14.418/14.648/0.230 ms
```

L'adresse IP destination `192.0.2.34` correspond à l'extrémité du tunnel à établir pour constituer la liaison VPN.

5. Le tunnel GRE entre routeurs d'extrémité

Dans cette section on configure les interfaces réseau de tunnel sur les routeurs TLS et GLA. Le premier est un routeur Cisco™ et le second un système Debian GNU/Linux. Les syntaxes de configuration entre les deux routeurs sont différentes mais les fonctionnalités sont vraiment très voisines.



Suite aux tests de la section précédente, on sait que les deux routeurs d'extrémité GLA et TLS se joignent via AMS. On configure donc une interface virtuelle de tunnel sur chacune des deux extrémités.

Le routeur GLA

Le paquet iproute fournit la commande **ip** qui contient les options de manipulation de tunnel. Cette commande et ces options sont appelées lors de l'initialisation de l'interface de tunnel sur le système Debian GNU/Linux. Voici l'extrait du fichier `/etc/network/interfaces` correspondant.

```
auto tunnel0
iface tunnel0 inet static
        address 198.51.100.2❶
        netmask 255.255.255.252
        pointopoint 198.51.100.1
        pre-up ip tunnel add tunnel0 mode gre local 192.0.2.66 remote 192.0.2.34 ttl 255 pmtudisc❷
        pre-up ip link set mtu 1440❷ dev tunnel0
        pre-up ip link set multicast on dev tunnel0
 post-up iptables -t mangle -A POSTROUTING -o tunnel0 -p tcp -m tcp --syn -j TCPMSS --set-mss 1400❸
 post-down iptables -t mangle -D POSTROUTING -o tunnel0 -p tcp -m tcp --syn -j TCPMSS --set-mss 1400
        post-down ip link set dev tunnel0 down
        post-down ip tunnel del tunnel0
```

Le routeur TLS

Sur le système IOS du routeur Cisco™, une interface de tunnel se configure directement. Voici l'extrait du fichier de configuration correspondant.

```
interface Tunnel0
 ip address 198.51.100.1❶ 255.255.255.252
 no ip redirects
 no ip proxy-arp
 ip mtu 1440❷
 ip virtual-reassembly in
 ip tcp adjust-mss 1400❸
 tunnel source GigabitEthernet0/0
 tunnel destination 192.0.2.66
 tunnel path-mtu-discovery❷
```

Les paramètres communs

❶e routage

Le réseau `198.51.100.0/30` est dédié au tunnel GRE. Une fois le tunnel en place les deux routeurs sont adjacents via ce réseau. Tous les paquets IP qui empruntent ce tunnel sont considérés comme ayant été directement transmis. L'utilisation de la commande **traceroute** sur les deux stations situées de part et d'autre du tunnel le montre bien.

```
etu@GLA-CLNT:~$ traceroute 10.11.0.2
traceroute to 10.11.0.2 (10.11.0.2), 30 hops max, 60 byte packets
 1  10.21.0.1 (10.21.0.1)  1.327 ms  1.309 ms  1.023 ms
 2  198.51.100.1 (198.51.100.1)  34.297 ms  14.241 ms  24.119 ms
 3  10.11.0.2 (10.11.0.2)  146.010 ms  166.488 ms  156.087 ms
```

```
etu@TLS-CLNT:~$ traceroute 10.21.0.2
traceroute to 10.21.0.2 (10.21.0.2), 30 hops max, 60 byte packets
 1  10.11.0.1 (10.11.0.1)  16.525 ms  6.306 ms  26.425 ms
 2  198.51.100.2 (198.51.100.2)  57.387 ms  47.014 ms  67.058 ms
 3  10.21.0.2 (10.21.0.2)  87.592 ms  77.351 ms  107.881 ms
```

Si le tunnel GRE n'était pas actif, on verrait apparaître un saut supplémentaire via le routeur AMS.

❷a taille maximum de paquet

La détection et/ou le réglage de la taille maximu de paquet ou Maximum Transmit Unit (MTU) est un point sensible dans la mise en œuvre d'un tunnel. L'utilisation du tunnel GRE génère un entête supplémentaire de 24 octets. Dans ce premier cas, le calcul est simple : MTU = 1500 - 24 = 1476. C'est avec l'ajout des protocoles IPsec que les choses se compliquent sérieusement. En effet le nombre des octets utilisés pour l'authentification avec le protocole ESP est difficile à évaluer. C'est la raison pour laquelle on applique généralement une dimension fixe par configuration à chaque extrémité du tunnel. Ici, la valeur **1440** a été appliquée sur les deux interfaces de tunnel.

```
etu@GLA-CLNT:~$ tracepath 10.11.0.2
 1:  10.21.0.2                                             0.177ms pmtu 1500
 1:  10.21.0.1                                             0.776ms
 1:  10.21.0.1                                             0.746ms
 2:  10.21.0.1                                             0.828ms pmtu 1440
 2:  198.51.100.1                                         14.643ms
 3:  10.11.0.2                                            15.260ms reached
     Resume: pmtu 1440 hops 3 back 62
```

En plus du paramétrage manuel, il est utile d'ajouter la fonction de découverte de la taille maximum de paquet le long du chemin entre les deux extrémités du tunnel. Il est possible qu'un réglage effectué par un opérateur tiers viennent «perturber» le schéma d'encapsulation initialement prévu. La commande **tracepath** fournie avec le paquet iputils-tracepath permet de contrôler la valeur MTU. La copie d'écran ci-dessus montre juste que les paramétrages implantés sur les deux interfaces de tunnel des routeurs d'extrémité sont effectifs.

Optimisation des connexions TCP

Compte tenu des multiples niveaux d'encapsulation, il est souvent nécessaire d'ajuster la taille maximum de segment lors du passage de la couche transport à la couche réseau avec le protocole TCP. Cette taille maximum de segment ou Maximum Segment Size (MSS) est un paramètre négocié lors de l'établissement de la connexion TCP avant l'échange de données.

Cette adaptation doit être appliquée à toute nouvelle connexion TCP engagée à travers le tunnel.

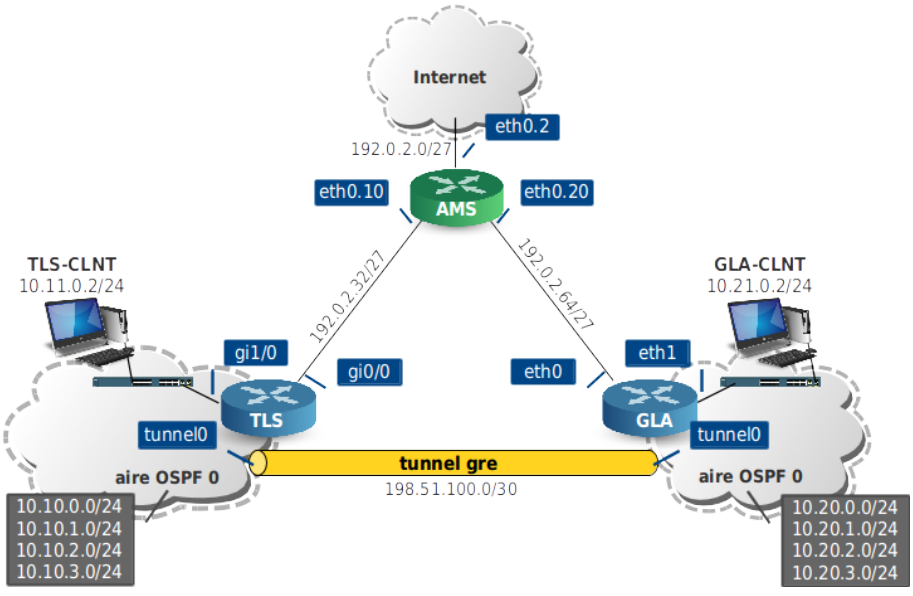
Coté routeur Cisco™, cet ajustement se fait au niveau de l'interface de tunnel avec la directive `ip tcp adjust-mss 1400`. La valeur `1400` tient compte d'une taille maximum d'en-tête TCP de 40 octets.

Côté GNU/Linux, il est nécessaire de faire appel aux outils de filtrage netfilter/iptables pour obtenir le même traitement. Dans l'exemple donné ci-dessus, on utilise la chaîne `POSTROUTING` de la table `mangle` dans laquelle on trouve tout ce qui concerne «l'altération» de paquet. Comme sur l'autre routeur, on applique la valeur `1400` au champ MSS pour toute nouvelle connexion TCP sortant par l'interface `tunnel0`.

Pour conclure cette section, le fichier `gre-tcp-iperf.pcap`[11] contient une capture de trafic traversant le tunnel GRE. Elle permet de visualiser les différentes encapsulations.

6. Le routage dynamique avec OSPF

Dans cette section on configure les démons du protocole de routage OSPF de façon à ce que les routeurs TLS et GLA échangent leurs informations de routage à travers le tunnel GRE configuré lors de l'étape précédente. Là encore les deux routeurs fonctionnent avec deux systèmes d'exploitation différents mais les caractéristiques du protocole OSPF restent les mêmes. C'est justement le propre d'un protocole !



[11] http://www.inetdoc.net/articles/site2site-ipsecvpn/files/gre-tcp-iperf.pcap

Suite aux tests des deux sections précédentes, on sait que les deux routeurs d'extrémité sont adjacents via le réseau IP du tunnel GRE.

Le routeur GLA

Sur le système Debian GNU/Linux, on utilise le démon OSPF fourni avec la suite logicielle quagga. On ne reprend pas ici les étapes d'installation et de configuration des démons. Les informations correspondantes sont données dans les documents Initiation au routage, 3ème partie[12] et Routage dynamique avec OSPF[13].

Les points de configuration essentiels sont les suivants.

- On commence par le paramétrage des bandes passantes pour chacun des liens dans zebra. Ces éléments sur les bandes passantes sont utilisés pour les calculs de métriques dans OSPF. Voici l'extrait du fichier `zebra.conf`[14].

```
!
interface dummy0
 bandwidth 8000000
 ipv6 nd suppress-ra
!
interface dummy1
 bandwidth 8000000
 ipv6 nd suppress-ra
!
interface dummy2
 bandwidth 8000000
 ipv6 nd suppress-ra
!
interface dummy3
 bandwidth 8000000
 ipv6 nd suppress-ra
!
interface eth0
 bandwidth 1000000
 ipv6 nd suppress-ra
!
interface eth1
 bandwidth 1000000
 ipv6 nd suppress-ra
!
interface tunnel0
 bandwidth 45000
 ipv6 nd suppress-ra
```

- On configure ensuite le démon OSPF. Voici l'extrait du fichier `ospfd.conf`[15].

```
!
interface tunnel0
 ip ospf network point-to-point
!
router ospf
 ospf router-id 0.0.0.20
 log-adjacency-changes
! Important: ensure reference bandwidth is consistent across all routers
 auto-cost reference-bandwidth 1000
 passive-interface default
 no passive-interface tunnel0
 network 10.20.0.0/16 area 0.0.0.0
 network 10.21.0.0/24 area 0.0.0.0
 network 198.51.100.0/30 area 0.0.0.0
```

[12] http://www.inetdoc.net/guides/zebra.ospf/
[13] http://www.inetdoc.net/travaux_pratiques/interco.ospf.qa/
[14] http://www.inetdoc.net/articles/site2site-ipsecvpn/files/gla-zebra.conf
[15] http://www.inetdoc.net/articles/site2site-ipsecvpn/files/gla-ospfd.conf

Le routeur TLS

Sur un routeur Cisco™, les interfaces et les démons de routage se configurent dans le même fichier de configuration. Les calculs de métriques héritent directement des paramètres fournis par les composants de pilotage des interfaces réseau. Voici l'extrait du fichier `TLS-confg`[16]

```
!
router ospf 1
 router-id 0.0.0.10
 auto-cost reference-bandwidth 1000
 passive-interface default
 no passive-interface Tunnel0
 network 10.10.0.0 0.0.255.255 area 0
 network 10.11.0.0 0.0.0.255 area 0
 network 198.51.100.0 0.0.0.3 area 0
```

Les tests de validation

Les étapes classiques de validation des échanges entre démons OSPF sont les suivantes.

- Les adjacences ; les routeurs doivent se reconnaître commen voisins.

```
GLA(ospfd)# sh ip ospf neighbor

    Neighbor ID Pri State           Dead Time Address         Interface            RXmtL RqstL DBsmL
0.0.0.10          1 Full/DROther      33.434s 198.51.100.1    tunnel0:198.51.100.2     0     0     0
```

```
TLS#sh ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
0.0.0.20          0   FULL/  -        00:00:35    198.51.100.2    Tunnel0
```

- Le tunnel GRE est vu par les deux routeurs comme un lien point à point.

```
GLA(ospfd)# sh ip ospf interface tunnel0
tunnel0 is up
  ifindex 11, MTU 1440 bytes, BW 45000 Kbit <UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>
  Internet Address 198.51.100.2/32, Peer 198.51.100.1, Area 0.0.0.0
  MTU mismatch detection:enabled
  Router ID 0.0.0.20, Network Type POINTOPOINT, Cost: 22
  Transmit Delay is 1 sec, State Point-To-Point, Priority 1
  No designated router on this network
  No backup designated router on this network
  Multicast group memberships: OSPFAllRouters
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    Hello due in 6.951s
  Neighbor Count is 1, Adjacent neighbor count is 1
```

```
TLS#sh ip ospf interface tunnel 0
Tunnel0 is up, line protocol is up
  Internet Address 198.51.100.1/30, Area 0, Attached via Network Statement
  Process ID 1, Router ID 0.0.0.10, Network Type POINT_TO_POINT, Cost: 22
  Topology-MTID    Cost    Disabled    Shutdown      Topology Name
        0           22        no          no            Base
  Transmit Delay is 1 sec, State POINT_TO_POINT
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:08
  Supports Link-local Signaling (LLS)
  Cisco NSF helper support enabled
  IETF NSF helper support enabled
  Index 6/6, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 4 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 0.0.0.20
  Suppress hello for 0 neighbor(s)
```

- Les routes annoncées sont bien échangées.

[16] http://www.inetdoc.net/articles/site2site-ipsecvpn/files/TLS-confg

```
GLA(ospfd)# sh ip ospf route
============ OSPF network routing table ============
N    10.10.0.1/32          [23] area: 0.0.0.0
                           via 198.51.100.1, tunnel0
N    10.10.1.1/32          [23] area: 0.0.0.0
                           via 198.51.100.1, tunnel0
N    10.10.2.1/32          [23] area: 0.0.0.0
                           via 198.51.100.1, tunnel0
N    10.10.3.1/32          [23] area: 0.0.0.0
                           via 198.51.100.1, tunnel0
N    10.11.0.0/24          [23] area: 0.0.0.0
                           via 198.51.100.1, tunnel0
N    10.20.0.0/24          [1] area: 0.0.0.0
                           directly attached to dummy0
N    10.20.1.0/24          [1] area: 0.0.0.0
                           directly attached to dummy1
N    10.20.2.0/24          [1] area: 0.0.0.0
                           directly attached to dummy2
N    10.20.3.0/24          [1] area: 0.0.0.0
                           directly attached to dummy3
N    10.21.0.0/24          [1] area: 0.0.0.0
                           directly attached to eth1
N    198.51.100.0/30       [44] area: 0.0.0.0
                           via 198.51.100.1, tunnel0
N    198.51.100.1/32       [22] area: 0.0.0.0
                           directly attached to tunnel0
```

```
TLS#sh ip ospf ro

            OSPF Router with ID (0.0.0.10) (Process ID 1)


                Base Topology (MTID 0)


    Area BACKBONE(0)

    Intra-area Route List
*   10.11.0.0/24, Intra, cost 1, area 0, Connected
      via 10.11.0.1, GigabitEthernet1/0
*>  10.20.0.0/24, Intra, cost 23, area 0
      via 198.51.100.2, Tunnel0
*>  10.20.1.0/24, Intra, cost 23, area 0
      via 198.51.100.2, Tunnel0
*>  10.20.2.0/24, Intra, cost 23, area 0
      via 198.51.100.2, Tunnel0
*>  10.20.3.0/24, Intra, cost 23, area 0
      via 198.51.100.2, Tunnel0
*>  10.21.0.0/24, Intra, cost 23, area 0
      via 198.51.100.2, Tunnel0
*   198.51.100.0/30, Intra, cost 22, area 0, Connected
      via 198.51.100.1, Tunnel0
*   10.10.0.1/32, Intra, cost 1, area 0, Connected
      via 10.10.0.1, Loopback0
*   10.10.1.1/32, Intra, cost 1, area 0, Connected
      via 10.10.1.1, Loopback1
*   10.10.2.1/32, Intra, cost 1, area 0, Connected
      via 10.10.2.1, Loopback2
*   10.10.3.1/32, Intra, cost 1, area 0, Connected
      via 10.10.3.1, Loopback3
*   198.51.100.1/32, Intra, cost 44, area 0
      via 198.51.100.2, Tunnel0
```

Note

La commande **sh ip ospf ro** fait partie des fameuses commandes cachées et non documentées de l'IOS.

7. La configuration IPsec

Avec cette section, on touche enfin au but ! Tous les autres éléments de la topologie type présentée en début de document sont en place et leur bon fonctionnement à été validé. Il ne reste plus qu'à appliquer la configuration IPsec choisie elle aussi en début de document.

On configure IPsec en mode transport avec le protocole ESP entre les deux extrémités du tunnel GRE.

Comme avec le tunnel GRE et le protocole OSPF, il est possible de faire correspondre au plus près les paramètres de configuration IPsec entre les deux routeurs utilisant des systèmes d'exploitation différents.

Côté Cisco™, on fait le choix de la configuration appelée `crypto map` et côté Debian GNU/Linux on utilise le couple de paquets ipsec-tools / racoon. L'intérêt de ce choix réside dans le fait que les associations de sécurité entre les deux routeurs sont négociées entre interfaces «physiques» tandis que le trafic protégé est acheminé via les interfaces «virtuelles» de tunnel.

La configuration IPsec du routeur GLA

Le paquet ipsec-tools fournit l'outil **setkey** qui sert à configurer les politiques de sécurité. Le fichier de configuration système correspondant est `/etc/ipsec-tools.conf`. En voici une copie.

```
#!/usr/sbin/setkey -f
#
## Flush the SAD and SPD
flush;
spdflush;

spdadd 192.0.2.34 192.0.2.66 any -P in ipsec
  esp/transport//require;

spdadd 192.0.2.66 192.0.2.34 any -P out ipsec
  esp/transport//require;
```

Avec les options retenues ci-dessus, on reprend les choix effectués en amont.

- On choisit adresses IP entre lesquelles les associations de sécurité sont établies pour chaque sens de trafic : entrant (`in`) ou sortant (`out`). Ces adresses IP correspondent aux interfaces physiques des deux routeurs. Voir le schéma de la topologie type.

- Le protocole ESP a été retenu après étude de l'encapsulation. Voir le graphique sur L'encapsulation GRE + IPsec. Cette représentation montre que le protocole AH ne présente aucun intérêt dans ce contexte puisque la totalité du paquet (en-tête + charge utile) est «traité» par ESP lorsqu'il est encapsulé dans un tunnel GRE. Le protocole ESP assure à lui seul les trois fonctions de la triade CIA.

- Le mode transport a aussi été choisi après étude de l'encapsulation. Le mode tunnel a été éliminé parce qu'il ne peut acheminer que du trafic unicast alors que l'on utilise un protocole de routage dynamique et que tous les échanges entre instances de routage se font à l'aide de trafic multicast. De plus, dans notre contexte, les extrémités de tunnel GRE coïncident avec les extrémités d'association de sécurité IPsec.

Le paquet racoon fournit un démon qui gère toutes les phases d'échanges de clé IKE. Le paramètrage des phases IKEv1 est défini dans le fichier système `/etc/racoon/racoon.conf` dont voici une copie.

```
#
# Please read racoon.conf(5) for details, and read also setkey(8).
#
log info;
path pre_shared_key "/etc/racoon/psk.txt";

padding {
  strict_check on;
}

listen {
  isakmp 192.0.2.66;
}

remote 192.0.2.34 {
  my_identifier address 192.0.2.66;
  exchange_mode main;
  proposal_check obey;

  proposal {
    lifetime time 86400 secs;
    encryption_algorithm aes 256;
    hash_algorithm md5;
    authentication_method pre_shared_key;
    dh_group 14;
  }

  generate_policy on;
  initial_contact on;
}

sainfo anonymous {
  lifetime time 3600 secs;
  pfs_group 14;
  encryption_algorithm aes 256;
  authentication_algorithm hmac_md5;
  compression_algorithm deflate;
}
```

Le choix des différents algorithmes est certainement le point le plus délicat de la configuration entre systèmes hétérogènes. Il faut rechercher «l'intersection» entre les critères de bonnes pratiques en matière de sécurité et les contraintes d'interopérabilité entre systèmes hétérogènes. La copie de fichier ci-dessus reflète bien le compromis adopté avec des points positifs et des point négatifs.

La phase 1 IKE

- Point négatif : `proposal_check obey;`

  Il aurait été préférable de pouvoir faire une correspondance exacte dans la négociation de la phase 1 IKE. Or, la version utilisée du paquet ipsec-tools ne supporte pas le paramètre `lifebyte`. Le compromis consiste donc à admettre les paramètres proposés par le routeur Cisco™.

- Point positif : `encryption_algorithm aes 256;`

  Les deux systèmes permettent de suivre les recommandations sur l'utilisation de l'agorithme de chiffrement AES. Le volet confidentialité est donc correctement traité (lettre **C** de la triade CIA). Voir le document RFC 4722.

- Point négatif : `hash_algorithm md5;`

  Avec ce choix d'algorithme de hachage et authentification, on privilégie la charge CPU au détriment de la sécurité. En effet, l'algorithme SHA-256 fait partie des recommandations actuelles alors que MD5 n'est plus considéré comme totalement sûr. Ce dernier algorithme consomme cependant nettement moins de cycles processeur.

  Du point de vue interopérabilité, tous les tests effectués avec les algorithmes SHA en phase 1 IKE ont échoué lors de la préparation de ce document. Avec la version de racoon utilisée, le premier tunnel de protection des échanges de clés ne s'établit pas et toute association de sécurité devient impossible à mettre en œuvre.

- Point négatif : `authentication_method pre_shared_key;`

  Ce choix de méthode d'authentification suppose que la clé secrète doit rester secrète. En effet, le niveau de sécurité ne dépend pas des algorithmes mais du secret qui sert à initier leurs traitements. Dans le contexte de ce document, le même secret est déposé manuellement sur les deux routeurs d'extrémité. Si un de ces équipement venait à être volé ou compromis, le secret le serait aussi.

  Il serait donc préférable d'utiliser un couple clé publique / clé privée propre à chaque extrémité. Une autorité de certification (CA) jouerait le rôle de tiers de confiance et permettrait de s'assurer de l'identité des équipements en communication. Dans notre cas, le VPN est établi entre les routeurs de deux sites distants. La sécurité repose donc sur la solidité des systèmes d'exploitation et sur la sécurisation des accès aux locaux techniques.

- Point positif : `dh_group 14;`

  L'échange de clé Diffie-Hellman utilise des groupes de 2048 bits pour protéger les échanges entre les deux extrémité. Cette configuration suit les recommandations en vigueur.

La phase 2 IKE

8. Pour conclure

Cet article est une illustration de quelques notions de routage avancé avec quatre instances de machines virtuelles. Il peut paraître un peu long ; comme la plupart des documents publiés sur inetdoc.net. Il se distingue cependant par le fait qu'à chaque élément de configuration avancé, on précise la démarche suivie pour caractériser la fonction implantée et on donne un extrait des résultats obtenus.

Même s'il est question de routage «avancé», la section sur la répartition de trafic illustre parfaitement le cours sur les modélisations. On caractérise le fait que chaque paquet IP peut suivre un chemin propre au niveau réseau tout en préservant une connexion de bout en bout au niveau transport. On retrouve ainsi les notions de réseau à commutation de paquets et de protocole orienté connexion.

Le marquage de paquets et l'enregistrement des communications sont aussi très intéressants et montrent qu'il est possible d'utiliser ces mécanismes sans nécessairement avoir recours à la traduction d'adresses. Sujet au combien polémique.

Enfin, l'architecture étudiée n'est qu'un exemple de ce qu'il est possible de faire avec des systèmes GNU/Linux. Il existe bien d'autres solutions ou variations pour aboutir au même résultat !

9. Les documents de référence

Voici une liste de liens vers les principaux documents utilisés pour construire cet article. En vis-à-vis de chaque lien, on trouve le(s) «point(s) essentiel(s)».

- An Illustrated Guide to IPsec[17] : Représentations graphiques détaillées sur les différentes encapsulation IPsec.
- Guide to IPsec VPNs[18] : Présentations des phases IKE et analyse de l'encapsulation avec WireShark. Étude de cas pratiques avec exemples de configurations ; utilisation de isakmpd notamment.
- Point-to-Point GRE over IPsec Design Guide[19] : Argumentation sur le choix entre les modes tunnel et transport.

[17] http://www.unixwiz.net/techtips/iguide-ipsec.html
[18] http://csrc.nist.gov/publications/nistpubs/800-77/sp800-77.pdf
[19] http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/P2P_GRE_IPSec/P2P_GRE.html

A. Configuration des interfaces

Pour les systèmes Debian GNU/Linux, on donne ci-dessous une copie du ficher `/etc/network/interfaces` tandis que pour le routeur Cisco™ on donne un extrait du fichier de configuration.

Routeur `AMS`

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet manual
        ip link set dev eth0 up

auto eth0.2
iface eth0.2 inet static
        address 192.0.2.2
        netmask 255.255.255.224
        network 192.0.2.0
        broadcast 192.0.2.31
        gateway 192.0.2.1

auto eth0.10
iface eth0.10 inet static
        address 192.0.2.33
        netmask 255.255.255.224
        network 192.0.2.32
        broadcast 192.0.2.63
        post-up ip route add 10.10.0.0/22 via 192.0.2.34
        post-up ip route add 10.11.0.0/24 via 192.0.2.34

auto eth0.20
iface eth0.20 inet static
        address 192.0.2.65
        netmask 255.255.255.224
        network 192.0.2.64
        broadcast 192.0.2.95
        post-up ip route add 10.20.0.0/22 via 192.0.2.66
        post-up ip route add 10.21.0.0/24 via 192.0.2.66
```

Routeur GLA

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
        address 192.0.2.66
        netmask 255.255.255.224
        network 192.0.2.64
        broadcast 192.0.2.95
        gateway 192.0.2.65

auto eth1
iface eth1 inet static
        address 10.21.0.1
        netmask 255.255.255.0
        network 10.21.0.0
        broadcast 10.21.0.255

auto dummy0
iface dummy0 inet static
        address 10.20.0.1
        netmask 255.255.255.0
        network 10.20.0.0
        broadcast 10.20.0.255
        pre-up modprobe dummy numdummies=4

auto dummy1
iface dummy1 inet static
        address 10.20.1.1
        netmask 255.255.255.0
        network 10.20.1.0
        broadcast 10.20.1.255

auto dummy2
iface dummy2 inet static
        address 10.20.2.1
        netmask 255.255.255.0
        network 10.20.2.0
        broadcast 10.20.2.255

auto dummy3
iface dummy3 inet static
        address 10.20.3.1
        netmask 255.255.255.0
        network 10.20.3.0
        broadcast 10.20.3.255
```

Routeur TLS

```
!
interface Loopback0
 ip address 10.10.0.1 255.255.255.0
!
interface Loopback1
 ip address 10.10.1.1 255.255.255.0
!
interface Loopback2
 ip address 10.10.2.1 255.255.255.0
!
interface Loopback3
 ip address 10.10.3.1 255.255.255.0
!
interface Ethernet0/0
 no ip address
 shutdown
 duplex auto
!
interface GigabitEthernet0/0
 ip address 192.0.2.34 255.255.255.224
 no ip redirects
 no ip proxy-arp
 ip virtual-reassembly in
 duplex full
 speed 1000
 media-type gbic
 negotiation auto
!
interface GigabitEthernet1/0
 ip address 10.11.0.1 255.255.255.0
 no ip redirects
 no ip proxy-arp
 ip virtual-reassembly in
 negotiation auto
```

B. Configuration de la commutation

Les interfaces réseau des instances de systèmes virtualisés sont un brassées sur une instance de commutateur, virtuel lui aussi, vde. Ce commutateur est fourni par le paquet `vde2` et il est lancé lors de l'initialisation de l'interface `tap0` sur le système hôte. Voici un extrait du fichier `/etc/network/interfaces`.

```
auto tap0
iface tap0 inet static
        address 192.0.2.1
        netmask 255.255.255.192
        network 192.0.2.0
        broadcast 192.0.2.63
        vde2-switch -
```

Tableau B.1. Brassage commutateur virtuel

Port VDE	Hôte	Interface(s)	Liaison
1	Système hôte	tap0	commutateur vde
2	ISP	eth0	système hôte | Internet
3	ISP	eth1.101	link101
		eth1.103	link103
4	R1	eth0.101	link101
		eth0.13	trunk R1 + R3
		eth0.12	trunk R1 + R2
5	R2	eth0	host
		eth0.12	trunk R2 + R1
		eth0.23	trunk R2 + R3
6	R3	eth0.103	link103
		eth0.13	trunk R3 + R1
		eth0.23	trunk R3 + R2
7	host	eth0	R2

Le fichier de configuration à charger au lancement du commutateur se présente comme suit.