Mobilité et Sécurité sur le réseau Réaumur, mise en place de solutions DHCP et VPN

Rapport de stage de licence R&T

Université Bordeaux 1
Service Réaumur
351 crs de la Libération
33400 Talence

Tuteur de stage:
Laurent FACQ

Stagiaire:
Pierre LEONARD
13 mars – 16 juin 2006

Responsable pédagogique:
Christophe BAILLOT

Remerciements

Je tiens tout d'abord à remercier mon tuteur de stage Laurent FACQ, pour son aide et ses conseils enrichissants qui m'ont permis d'avancer intelligemment dans ma réflexion et mon travail. Son savoir aura été une mine de renseignements pendant les 3 mois qu'aura duré ce stage.

Je tiens aussi à remercier l'ensemble du personnel Réaumur [1] pour m'avoir accueilli en son sein et m'avoir permis d'effectuer ce stage dans les meilleures conditions qui soient. Les différents contacts humain et professionnel auront été très fructueux pour moi.

[1] Réseau Aquitain des Utilisateurs en Milieu Universitaire et de Recherche

Table des matières

Annexes

Table des figures

1 Présentation du service Réaumur

1.1 Origine

Le réseau Rénater [2] a été créé en 1991 dans le but d'interconnecter les différentes universités et laboratoires de recherche français via des liaisons dédiées et spécialisées ainsi que de les relier à Internet.

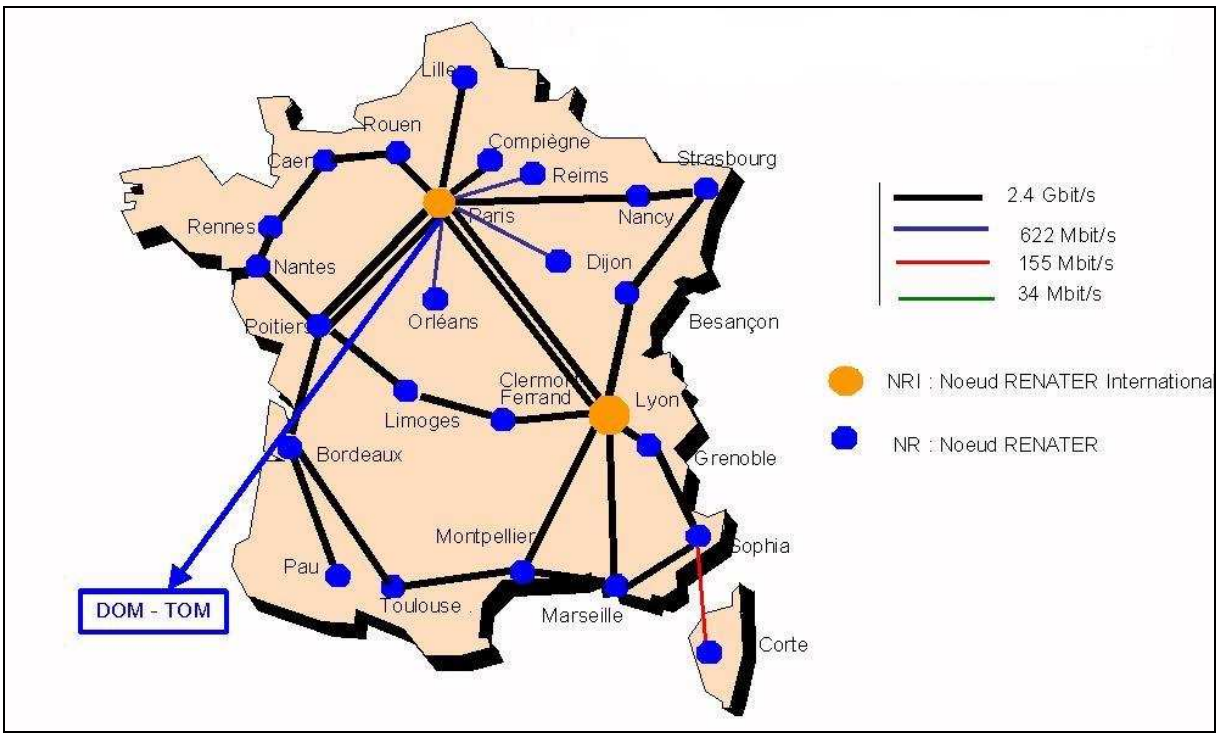


Figure 1-1 : Topologie du réseau Rénater

Le service Réaumur est né d'un besoin de créer un service régional d'interconnexion des différents organismes universitaires Aquitains.

1.2 Missions

Le service Réaumur est un service inter-universitaire qui a été créé afin de prendre en charge le réseau informatique de campus à haut débit. Une de ses missions pour les partenaires du campus Talence, Pessac, Gradignan est d'assurer la maintenance et le développement de moyens et de services réseaux communs dans le cadre des enseignements universitaires et de la recherche. Réaumur fixe notamment les conditions sécuritaires d'utilisation, en accord avec les partenaires. Le service de base réalisé par Réaumur consiste à mutualiser les services de connexion à Rénater. Réaumur a également une mission au niveau régional, puisqu'il est gestionnaire de la plaque réseau régionale ESRA pour les établissements d'**E**nseignement **S**upérieur et de **R**echerche en **A**quitaine. Cette mission concerne les partenaires déjà cités auxquels s'ajoutent le Rectorat de Bordeaux, le CEMAGREF, l'EAPBx, l'ENITA de Bordeaux, l'INRA Aquitaine, l'IUFM d'Aquitaine, l'Université de Pau et des Pays de l'Adour, et enfin l'Ecole de Management de Bordeaux.

[2] Réseau National de télécommunications pour la Technologie l'Enseignement et la Recherche

1.3 Personnel

Le service Réaumur est composé de 8 personnes + 2 stagiaires actuellement:

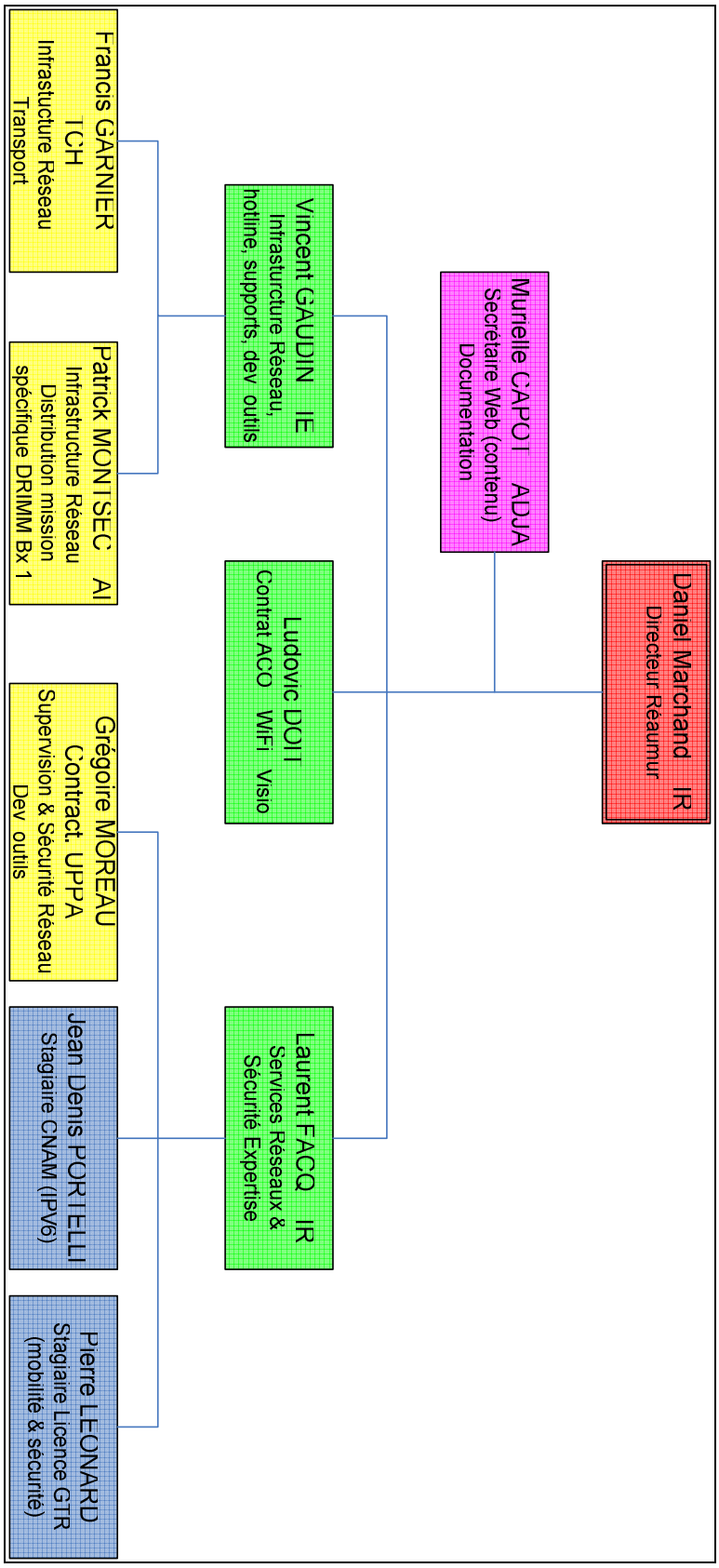


Figure 1-2 : Organigramme

1.4 Topologie du réseau Réaumur

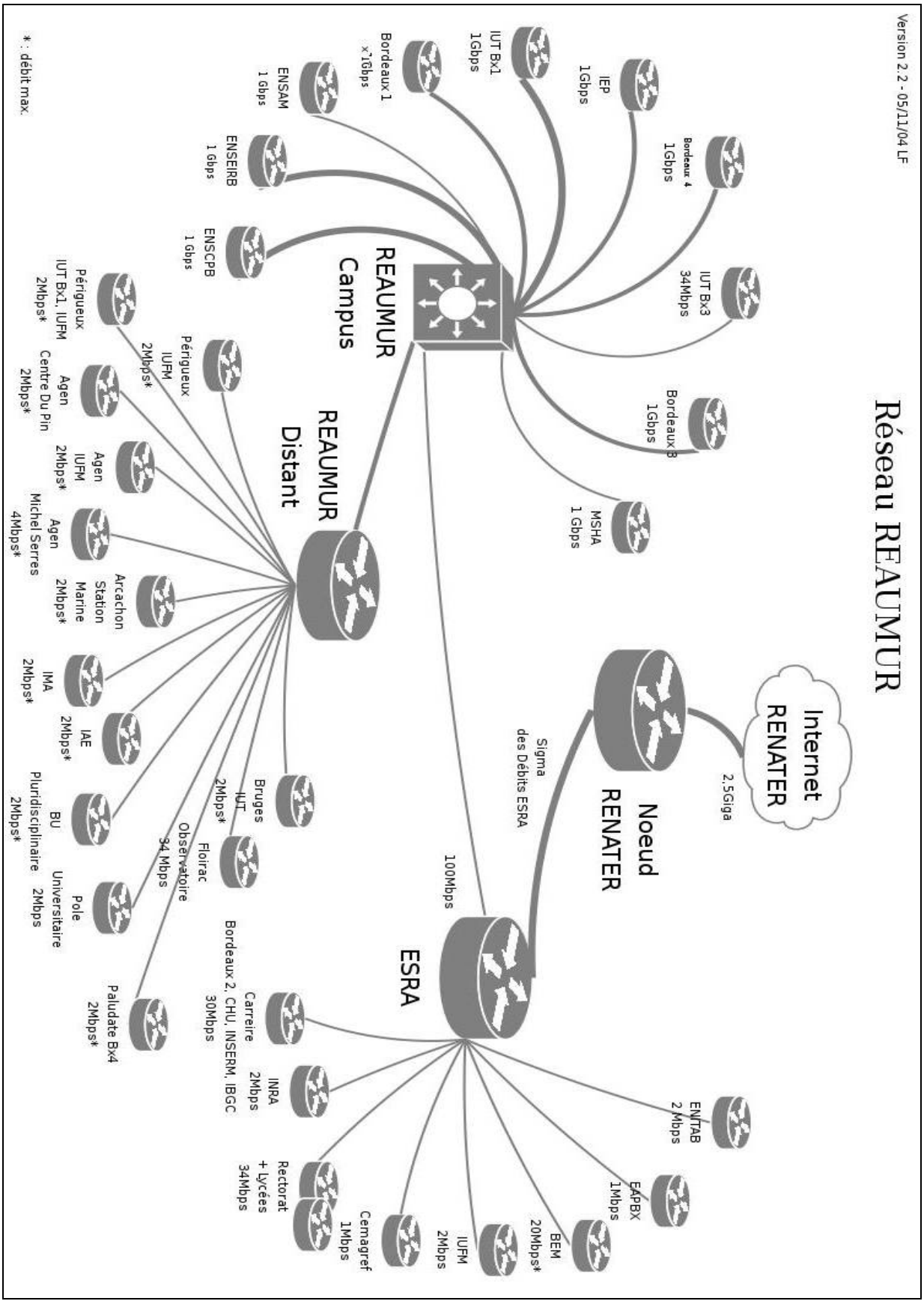


Figure 1-3 : Topologie du réseau Réaumur

2 Présentation du sujet et de son contexte

2.1 Introduction

Le réseau universitaire de Bordeaux est très vaste et nécessite de pouvoir authentifier et tracer tous les utilisateurs quelque soit leur mode de connexion sur celui-ci, pour cela chaque cas doit être étudié et une solution adapté envisagée. Il existe, sur le réseau du campus, plusieurs cas de connexion pouvant poser des problèmes de sécurité et/ou d'authentification tels que les congrès ou les réunions avec des intervenants extérieurs ayant besoin d'accès au réseau, les accès distants ou certains studios possédant des accès au réseau, etc... Il faut donc être en mesure d'apporter des solutions à toutes ces situations, et ce, de manière sécurisé. Nous allons nous intéresser ici à deux cas particuliers que sont les serveurs DHCP avec attribution d'adresse IP fixe en fonction du port, ainsi que les accès distants via VPN. Ces différents contrôles d'accès dépendent du lieu, de la population visé ainsi que du matériel utilisé.

L'installation de serveurs DHCP [3] s'inscrit dans le cadre d'une ouverture et d'une simplification de l'utilisation du réseau Rénater et Internet. Mais ce mode d'attribution d'adressage implique un manque de suivi des connexions donc un amoindrissement de la sécurisation des réseaux informatiques et de la prévention des actes illicites, c'est pourquoi il est très important de nos jours de pouvoir maintenir un niveau de sécurité suffisant et nécessaire au sein d'un réseau tel que le réseau Réaumur, ou à plus grande échelle Rénater.

Cette maintenance de sécurité et de suivi passe par une bonne configuration de tous les équipements du réseau et notamment par la configuration de serveurs DHCP, en charge de l'attribution dynamique d'adresse IP:

- Fixé : dans le cas d'attribution en fonction de l'adresse MAC [5] de la carte réseau de l'utilisateur ou de la position de l'utilisateur sur le réseau
- Aléatoire : pour des utilisateurs totalement nomade au sein du réseau du campus recevant des adresses appartenant à un pool d'adresses à disposition et géré par le serveur.

En ce qui concerne la solution VPN [4], son but est d'assurer la confidentialité des échanges entre un utilisateur connecté depuis l'extérieur et le réseau universitaire via un tunnel chiffré et authentifié de part et d'autre, le tout via le media non sûr qu'est l'Internet. Ce type de solution est bien moins cher que la location de lignes spécialisées, bien que ces dernières assurent un débit maximal et une qualité de service supérieure à celle d'un VPN.

2.2 Contexte

Le campus possède des studios mis à la disposition enseignants/chercheurs étrangers de passage, et équipés de prises Ethernet pour la connexion au Réseau du campus.

Dans la configuration actuelle de cette partie du réseau, les utilisateurs de ces chambres ont la possibilité de se connecter à l'Internet à l'aide d'une configuration IP fixe affichée sur la porte du studio. Chaque prise est ensuite filtrée par des ACLs [6] configurés sur les équipements en charge de la connexion de ces studios sur le réseau universitaire. Cette méthode permet de pouvoir tracer un utilisateur puisqu'à une IP correspond un studio et, à un studio et une date correspond un locataire.

[3] Dynamic Host Configuration Protocol
[4] Virtual Private Network
[5] Media Access Control
[6] Access Control List

Mais cette gestion de l'adressage pose plusieurs problèmes puisque les utilisateurs novices en matière de réseau informatique, et principalement en ce qui concerne leur configuration, se trouvent confronté à une situation dans laquelle ils doivent eux même configurer leur machine pour accéder au réseau :

- Mauvaise connaissance des réseaux informatiques
- Méconnaissance des zones de configuration de leur machine
- Mauvaise lecture de l'adressage imposé dans le studio où ils se trouvent
- Mauvaise configuration de la machine
- Surcharge de travail pour le personnel du Réaumur

Résultat => Peu pratique et gourmand en assistance

La solution envisagée est donc d'installer un serveur DHCP qui assignera automatiquement la bonne IP à chacun des utilisateurs de ces studios, un peu comme un FAI [7] le ferait. La configuration en IP fixe en fonction de l'adresse MAC de la machine n'est pas envisageable dans l'état actuel des choses puisque la propriété principale d'une adresse MAC est justement son unicité or le locataire peut changer à tous moment. Une des configurations possibles consiste à utiliser l'option 82 du serveur DHCP, il s'agit du "Relay Information Option", en d'autres termes le champ option de la requête DHCP va être complété par l'agent de relais (routeur ou switch); cette option permettra ensuite de retracer la provenance d'une requête DHCP.

Il existe également un projet d'accès distants mettant en œuvre des VPN et permettant une fois l'authentification effectuée, de placer l'utilisateur dans son réseau de travail d'origine puisque chaque service ou site correspond à un VLAN sur le réseau Réaumur.
Il existe pour cela deux solutions différentes à étudier:

- Openvpn : ce logiciel provenant du monde libre s'installe sur un système d'exploitation de type Unix et fonctionne sur un modèle client/serveur. Il permet d'assurer la communication chiffrée, grâce au protocole SSL/TLS, et authentifiée à l'aide de certificats et/ou d'un système de clés partagées.
- Utiliser un routeur en tant que concentrateur VPN; il faut pour cela mettre à jour l'IOS[8] du routeur en charge de concentrer les VPN, voir même lui ajouter un module d'accélération de cryptologie, dans le but de pouvoir créer simultanément un grand nombre de tunnels chiffrés établis à l'aide du protocole IPSec assurant la confidentialité des échanges.

[7] Fournisseur d'Accès Internet
[8] Internet Operating System

3 Solution DHCP

3.1 Présentation du protocole

3.1.1 Introduction

Le protocole DHCP est un protocole de configuration dynamique sur un réseau via le protocole UDP [9]. Il permet entre autre d'assigner une adresse IP aux différentes machines, mais aussi de leur fournir les informations nécessaires au bon fonctionnement du réseau (passerelle, serveurs DNS, domaine…) et tout ça dynamiquement sans que l'utilisateur n'est rien d'autre à faire que de se relier physiquement au réseau et d'activer l'option DHCP.

Il s'agit d'un protocole client/serveur défini par les RFCs [10] 2131 et 2132 et basé sur le protocole BOOTP [11], apportant à celui-ci de nouvelles fonctionnalités comme l'allocation dynamique d'adresses IP : il n'est plus utile de renseigner manuellement une table de correspondance adresse MAC / adresse IP. Une adresse IP peut être attribuée à partir d'un ou plusieurs pools d'adresses IP disponibles. En effet, le DHCP permet à l'administrateur du réseau de créer un ensemble d'adresses pouvant être attribuées aux clients. De cette façon, il peut déterminer des intervalles d'adresses IP. Il peut donc décider de donner un intervalle pour un segment et un autre intervalle pour un deuxième segment. De cette façon, la gestion du réseau est simplifiée. Le protocole BOOTP ne permet pas cette possibilité.

Pour des raisons d'optimisation des ressources réseau, DHCP permet l'allocation d'une adresse IP pour une certaine période de temps c'est ce qu'on appel un bail (lease en anglais), les adresses IP sont délivrées avec une durée de validité. Cette caractéristique du DHCP lui permet de garder en mémoire, pour une certaine période de temps, les adresses IP qu'il assigne aux clients, ainsi que les informations de l'ordinateur en question, cela entraîne donc moins de transferts lors d'une connexion. La période de temps est définie par l'administrateur du réseau. Le protocole BOOTP, quant à lui, n'a pas cette fonction; les postes clients utiliseront donc leur adresse IP et les informations reçus sur la configuration jusqu'à ce qu'il soit redémarré.

Le protocole DHCP est surtout considéré comme un remplacent du BOOTP. La plupart des implémentations TCP/IP [12] moderne offre aux clients le protocole DHCP à l'aide d'un serveur DHCP.

Le serveur DHCP utilise 2 ports :

- Port serveur : 67
- Port client : 68

C'est-à-dire, par exemple, que les requêtes vont du port UDP 68 du client vers le port UDP 67 du serveur.

[9] User Datagram Protocol
[10] Request For Comment
[11] Bootpstrap Protocol
[12] Transmission Control Protocol / Internet Protocol

3.1.2 La trame DHCP

Voici le contenu d'une trame DHCP :

Type du Message : **op** (1)	Type de l'adresse MAC : **htype** (1)	Longueur de l'adresse MAC : **hlen** (1)	Nombre de saut : **hops** (1)
Identifiant de la transaction choisi aléatoirement : **xid** (4)			
Temps écoulé depuis le début de la transaction : **secs** (2)		Flag de broadcast : **flag** (2)	
Adresse IP du client renseignée uniquement en cas de statut Bound, Renew, Rebinding : **ciaddr** (4)			
Adresse IP du client renvoyée par le serveur DHCP : **yiaddr** (4)			
Adresse IP du serveur à utiliser dans la prochaine étape du processus Bootp : **siaddr** (4)			
Adresse IP de l'agent de relais DHCP : **giaddr** (4)			
Adresse MAC du client : **chaddr** (16)			
Adresse optionnelle d'un serveur : **sname** (64)			
Nom du fichier à utiliser pour le boot : **file** (128)			
Options : **options** (variable)			

* Les paramètres entre parenthèses indiquent la taille du champ en octets.

Quelques précisions sur certains champs :

- **op :** est le type de message, si op vaut 1 le message est un DHCPRequest (trame DHCP émise par le client à destination du serveur) si op vaut 2 c'est un DHCPReply (trame DHCP émise par le serveur à destination du client).

- **htype :** défini le type de l'adresse MAC (ex : 1=Ethernet 10Mb/s).

- **xid :** est l'identifiant de la transaction. Utilisé pour associer les requêtes d'un client et les réponses d'un serveur à une même transaction DHCP, il est choisi par le client DHCP et doit être unique sur le réseau local.

- **flags :** Le bit positionné le plus à gauche de ce champ est appelé le "Flag de Broadcast". Certains clients DHCP ne sont pas capables de recevoir des datagrammes unicast avant que leur pile TC/IP ne soit configurée. Dans ce cas, le client DHCP positionne à 1 le "Flag de Broadcast" (le bit le plus à gauche du champ flags) sur toute trame DHCP qu'il émet. Un serveur DHCP recevant une telle trame répond alors systématiquement par broadcast. Si le "Flag de Broadcast" n'est pas positionné, les réponses des serveurs peuvent être émises sous forme d'unicast, l'adresse MAC destinatrice étant celle du champ chaddr lu dans la trame cliente et l'adresse IP destinatrice étant celle offerte par le serveur.

- **siaddr :** Adresse IP du serveur à utiliser dans la prochaine étape du processus Bootp. Ce champ est renseigné dans les messages DHCP Offer et DHCP Ack.

3.2 Fonctionnement du protocole

3.2.1 Illustration

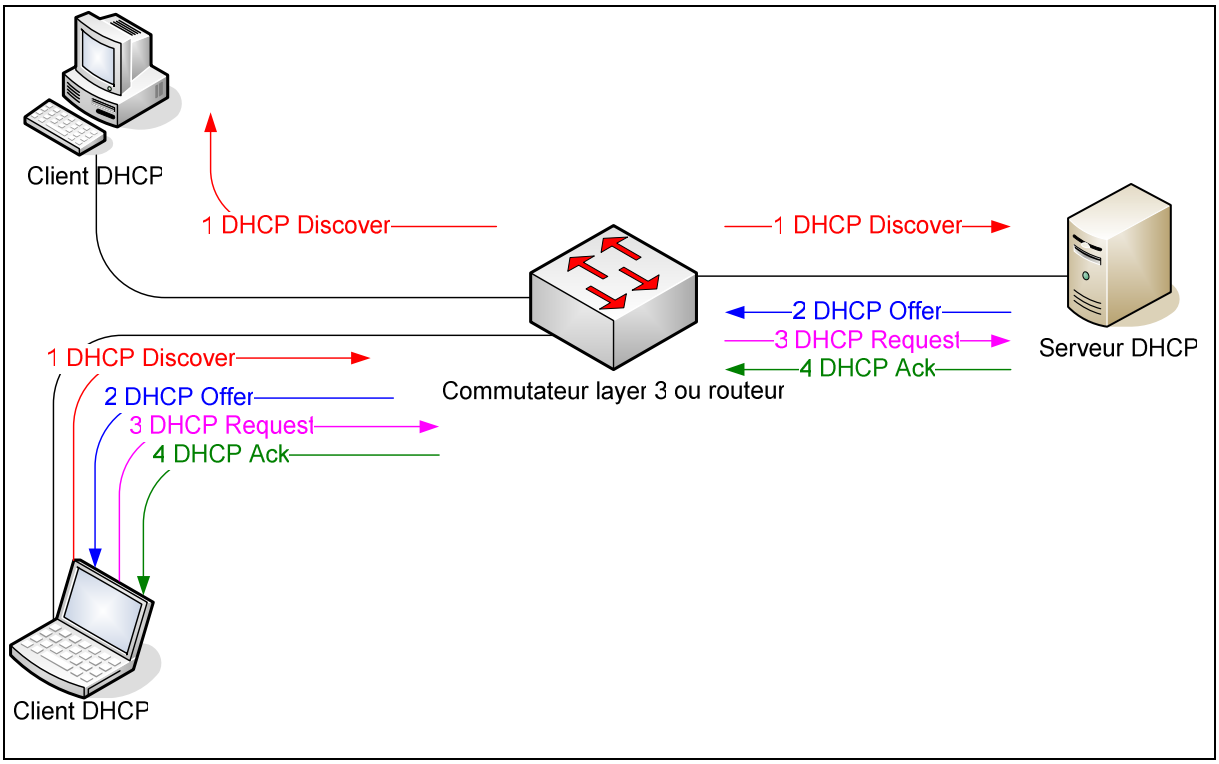


Figure 3-1 : Fonctionnement DHCP

3.2.2 Explication

Client	Serveur
1 DHCPDISCOVER Le client recherche les serveurs DHCP disponibles	
	2 DHCPOFFER Le serveur dit qu'il est prêt à attribuer une adresse IP
3 DHCPREQUEST Le client demande l'attribution de l'adresse IP	
	4 DHCPACK Le serveur DHCP confirme l'attribution de l'adresse IP et envoies des informations de configuration réseau diverses (DNS, passerelle, etc...)
...	
DHCPREQUEST Demande de renouvellement du bail	
	DHCPACK Le serveur attribue le bail au client
...	
DHCPRELEASE Le client n'a plus besoin d'accéder au réseau, il libère son adresse IP	
	Le serveur met fin au bail et libère l'adresse IP, cette dernière devient donc réutilisable

3.3 Option 82 : DHCP Relay Agent Information

La mise en œuvre d'une architecture telle que décrite dans le contexte du sujet de stage, nécessite d'approfondir la notion de DHCP. En effet il existe une option relativement récente (portant le numéro 82) permettant de retracer la provenance d'une trame DHCP sur le réseau, dans notre cas cela va grandement nous aider étant donné que le but est d'attribuer une adresse IP figée sur une « prise Ethernet donnée » et cela quelque soit l'utilisateur et la machine connecté derrière.

Pour mettre en œuvre cette option il est nécessaire de posséder sur le réseau des équipements possédant les fonctionnalités suivantes (Attention ! Il s'agit de fonctionnalité sur des équipements Cisco) :

- Ip dhcp snooping
- Ip dhcp relay information option

3.3.1 DHCP Snooping

Pour protéger au maximum le réseau des intrusions malveillantes il existe une fonctionnalité disponible sur des switchs tels que le Cisco catalyst 2950 Extended Image (utilisé pour la maquette) permettant de spécifier une interface de confiance sur laquelle seront émises des trames pouvant provenir d'un serveur DHCP, ainsi il sera impossible à un intrus d'installer un deuxième serveur DHCP pirate sur un autre port que celui définit de confiance. Cette option doit également être appliquée sur les vlans concernés par le serveur DHCP. Cette option conditionne l'utilisation de l'option suivante.

3.3.2 DHCP Relay Information Option

Cette fonctionnalité est la plus intéressante des deux puisque c'est elle qui va nous permettre de tracer une requête DHCP. En effet une fois activé, cette fonctionnalité permet de marquer les trames DHCPRequest en insérant dans le champ option de celle-ci, avec le code option 82, les informations suivantes :

- Numéro de VLAN [13] du port
- Numéro du module
- Numéro du port
- Adresse MAC

du switch sur lequel la trame arrive.

Il suffit alors simplement de récupérer ces informations et de les traiter du côté du serveur DHCP afin d'assigner une adresse IP figé en fonction de ces paramètres.

[13] Virtual Local Area Network

3.3.3 Structure de l'option 82

Code	Length	Agent Information Field					
82	N	i1	i2	i3	i4	…	iN

Cette option est divisée en deux sous options :

- **Agent Circuit ID :** cette sous option contient le vlan, le module et le port de l'interface sur laquelle la trame dhcp arrive et porte le code option 1.

Suboption type (1)	Length (1)	Circuit ID type (1)	Length (1)	Vlan (2)	Module (1)	Port (1)
01	06	00	04	XXXX	YY	ZZ

** Les paramètres entre parenthèses indiquent la taille du champ en octets.*

- **Agent Remote ID :** cette sous option contient l'adresse MAC du switch relayant la trame dhcp et porte le code option 2.

Suboption type (1)	Length (1)	Remote ID type (1)	Length (1)	MAC Address (6)
02	08	00	06	aa :bb :cc :dd :ee :ff

** Les paramètres entre parenthèses indiquent la taille du champ en octets.*

3.4 Architecture de la maquette

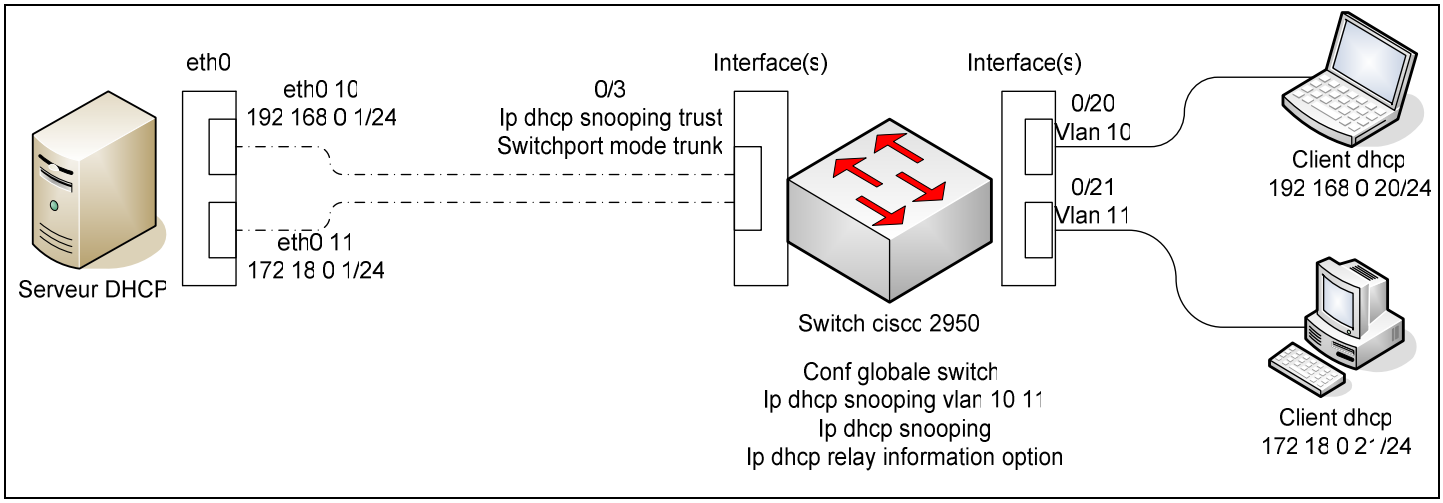


Figure 3-2 : Maquette DHCP

** Les configurations et informations indiquées sur le schéma sont les véritables données utilisées pour les tests.*

Remarque :

Pour les besoins de la maquette il a été nécessaire de créer des sous interfaces virtuelles sur l'interface physique du serveur DHCP dans le but de communiquer avec les vlans configurés sur le switch. Cette manipulation a été effectué à l'aide de l'outil vconfig sous Debian.

3.5 Développement du script PERL

3.5.1 Le script

Pour faciliter la génération du fichier de configurations dhcp, il m'a été demandé de développer un script en PERL [14] (joint en annexe).

Le cahier des charges était le suivant :

- Génération des fichiers de définition de l'adressage IP
- Génération du fichier de classes
- Génération des fichiers contenant la configuration des switchs mis en oeuvre.

Le but de ce script étant de générer tous ces fichiers il est nécessaire de récupérer les informations dans un autre fichier que l'on appellera fichier utilisateur. Ce fichier possède une syntaxe simple mais imposée afin que le script puisse l'interpréter correctement ; il suffit en fait de préciser derrière des mots clés les paramètres importants tels que le vlan, le module, le port, le nom et l'adresse mac du switch, le nom du sous réseau que l'on souhaite voir apparaître dans le fichier de configuration et le chemin d'enregistrement des fichiers finaux.
Mais avant tout il est essentiel de connaître la syntaxe d'un fichier de configuration DHCP: dhcpd.conf

3.5.2 Dhcpd.conf

3.5.2.1 Syntaxe

Comme tout fichier de configuration, le fichier de configuration du serveur dhcp à une syntaxe précise à respecter, voilà à quoi il doit ressembler pour une configuration simple :

```
paramètres globaux (serveur dns, durée du bail, nom de domaine…)

shared-network name {
  paramètres spécifiques au réseau partagé
        subnet 192.168.1.0 netmask 255.255.255.0 {
                paramètres spécifiques au sous-réseau.
                range 192.168.1.1 192.168.1.50;
         }
        subnet 192.168.2.0 netmask 255.255.255.0 {
                paramètres spécifiques au sous-réseau
                range 192.168.2.1 192.168.2.50;
        }
}
```

Il est aussi possible d'assigner une adresse IP fixe en fonction de l'adresse MAC de la machine avec la syntaxe :

```
host toto {
        hardware ethernet aa:bb:cc:dd:ee:ff ;
        fixed-address 192.168.0.10;
}
```

Pour ce genre de configuration on appellera cela du dhcp simple par opposition au dhcp évolué que nous allons voir maintenant.

[14] Practical Extraction and Report Language

3.5.2.2 Dhcp évolué

Il s'agit ici d'utiliser les paramètres de l'option 82 de la distribution DHCP de l'Internet Software Consortium, que le switch a complété dans le paquet DHCP, en fait il est presque question de "programmer" dans le fichier de configuration.
Pour utiliser ces paramètres il faut définir des classes vérifiant les conditions nécessaires à l'attribution d'une adresse IP figée. Il faut également savoir que pour des raisons pratiques les paramètres utilisés sont fournis en hexadécimal et avec des ":" pour séparer chaque octets.

Par exemple si on veut assigner l'adresse IP fixe 192.168.0.20 à n'importe quelle machine reliée au switch ayant l'adresse MAC aa:bb:cc:dd:ee:ff , sur le port 20 (interface du switch 19 + 1 chez cisco) du module 0 dans le vlan 10 on aura des paramètres qui seront alors en héxadécimal :

- Même adresse MAC
- Port : 14
- Module : 00
- Vlan : 00 :0a

La syntaxe à insérer en haut du fichier de configuration sera alors du type :

```
class "classTest"{
      match if (substring (option agent.circuit-id,2, 2) = 00:0a)
      and (substring (option agent.circuit-id,4,1) = 00)
      and (suffix (option agent.circuit-id, 1) = 14)
      and (suffix(option agent.remote-id,6) = aa:bb:cc:dd:ee:ff);
}
```

et celle à insérer dans la définition du subnet est :

```
shared-network name {
  paramètres spécifiques au réseau partagé
        subnet 192.168.0.0 netmask 255.255.255.0 {
            pool {
                  allow members of " classTest";
                  range 192.168.0.20;
            }
        }
}
```

Remarque : Les méthodes substring ( ) et suffix ( ) permettent de découper les champs des sous options circuit-id et remote-id de l'option agent.

Il existe aussi une directive "include" propre au fichier dhcpd.conf permettant d'inclure des fichiers indépendant, le but ici étant de générer ces fichiers via le script PERL et de les inclure ensuite. C'est-à-dire que l'utilisateur contrôle le fichier dhcpd.conf puis peut y inclure des fichiers de configuration supplémentaire limitant ainsi les conflits. Cela permet un travail moins fastidieux et plus souple dans la mesure où si le nombre de configurations similaires est important il y a de grands risques de faire des erreurs de syntaxes.

3.5.3 Utilisation du script et du fichier utilisateur

L'utilisation du script et du fichier de configuration utilisateur est relativement simple (pour cela un manuel a été écrit reprenant toutes les étapes décrites ci-dessous), il suffit de spécifier les paramètres derrière des directives.

- **directive ROOT :** elle permet de spécifier le chemin d'enregistrement de tous les fichiers et se présente sous la forme
  - ➔ ROOT <chemin absolu>

- **directive NETWORK :** elle permet de spécifier un nom qui servira de nom de fichier pour la déclaration des pools et apparaîtra également dans le nom de la classe correspondante
  - ➔ NETWORK <nom du network>

- **directive SWITCH :** elle permet de renseigner le nom, l'adresse mac et éventuellement le type du switch sur lequel sera branché l'utilisateur final à filtrer
  - ➔ SWITCH <nom du switch> <@mac> [<type du switch>]

  Remarque: Par défaut le type du switch est « cisco ».

- **directive VLAN :** comme son nom l'indique on y spécifie le numéro de vlan de l'utilisateur à filtrer
  - ➔ VLAN <numéro de vlan>

- **directive PORT :** cette directive est la plus complète de toute puisqu'on spécifie ici le module et le port du switch sur lequel est branché l'utilisateur à filtrer ainsi que l'adresse ip que l'on veut lui attribuer
  - ➔ PORT <numéro de module> <numéro de port> <@ip à attribuer> [<commentaire>]

Une fois exécuté, il aura généré plusieurs fichiers de configuration :

- autant de fichiers de définition de pool que de sous réseaux déclarés. Les noms de ces fichiers sont constitués du nom fournit au niveau de la directive NETWORK et se terminent par ".conf ".

- un fichiers global contenant les déclarations des diverses classes. Ce fichier porte le nom de "classes.conf ".

- 1 fichier par switch concerné et contenant sa configuration concernant les ACLs. Les fichiers de configuration des switchs porteront le nom du switch fournit à la suite de la directive SWITCH et se termineront par ".conf ".

Des exemples de tous ces fichiers sont fournis en annexe.

4 Solutions VPN

4.1 Notion de VPN

4.1.1 Introduction au VPN

Les applications et les systèmes distribués font de plus en plus partie intégrante du paysage d'un grand nombre d'entreprises. Ces technologies ont pu se développer grâce aux performances toujours plus importantes des réseaux locaux. Mais le succès de ces applications a fait aussi apparaître un de leur écueil. En effet si les applications distribuées deviennent le principal outil du système d'information de l'entreprise, comment assurer leur accès sécurisé au sein de structures parfois réparties sur de grandes distances géographiques ? Concrètement comment une succursale d'une entreprise peut-elle accéder de façon sécurisé aux données situées sur un serveur de la maison mère distant de plusieurs milliers de kilomètres comme si elles étaient locales ? Les VPN, Virtual Private Network ou RPV, Réseau Privé Virtuel en français, ont commencé à être mis en place pour répondre à ce type de problématique. Mais d'autres problématiques sont apparues et les VPN ont aujourd'hui pris une place importante dans les réseaux informatique et l'informatique distribuées.

4.1.2 Principe de fonctionnement

Un réseau VPN repose sur un protocole appelé "protocole de tunneling". Ce protocole permet de faire circuler les informations de l'entreprise généralement de façon cryptée, d'un bout à l'autre du tunnel; ainsi les utilisateurs ont l'impression de se connecter directement sur le réseau de l'entreprise. Le principe du tunneling consiste à construire un chemin virtuel après avoir identifié l'émetteur et le destinataire. Par la suite, la source chiffre les données au moyen d'algorithmes de cryptographie négociés entre le client et le serveur et les achemine en empruntant ce chemin virtuel. Afin d'assurer un accès aisé et peu coûteux aux intranets et extranets d'entreprise, les réseaux privés virtuels d'accès simulent un réseau privé alors qu'ils utilisent en réalité une infrastructure d'accès partagée telle que l'Internet. Les données à transmettre peuvent être prise en charge par un protocole différent d'IP, mais dans ce cas le protocole de tunneling encapsule les données en ajoutant un en-tête. Le tunneling est l'ensemble des processus d'encapsulation, de transmission et de désencapsulation.

4.1.3 Utilisation d'un VPN

Il existe trois types standard d'utilisation des VPNs:

- **Le VPN d'accès** qui est utilisé pour permettre à des utilisateurs itinérants d'accéder au réseau privé.
- **L'intranet VPN** qui permet de relier deux intranets distants d'une même entreprise.
- **L'extranet VPN** qui permet d'ouvrir une partie ou la totalité du réseau privé à un client ou un partenaire de l'entreprise afin de communiquer avec lui.

Un système de VPN doit pouvoir mettre en oeuvre les fonctionnalités suivantes :

- **Authentification d'utilisateur:** Seuls les utilisateurs autorisés doivent pouvoir s'identifier sur le réseau virtuel. De plus, un historique des connexions et des actions effectuées sur le réseau doit être conservé.

- **Gestion d'adresses:** Chaque client sur le réseau doit avoir une adresse propre. Cette adresse doit rester confidentielle. Un nouveau client doit pourvoir se connecter facilement au réseau et recevoir une adresse.

- **Cryptage des données:** Lors de leurs transports sur le réseau public les données doivent être protégées par un cryptage efficace.

- **Gestion de clés:** Les clés de cryptage pour le client et le serveur doivent pouvoir être générées et régénérées.

- **Prise en charge multiprotocole:** La solution VPN doit supporter les protocoles les plus utilisés sur les réseaux publics en particulier IP.

4.1.4 Protocole utilisé pour réaliser une connexion VPN

Il existe plusieurs classes de protocoles permettant de réaliser une connexion VPN:

- **Protocole de niveau 2** comme PPTP (Point to Point Tunneling Protocol) développé et soutenu de nos jours par Microsoft (notamment utilisé dans les systèmes d'exploitations Windows), L2F (Layer Two Forwarding) développé par Cisco et presque disparu aujourd'hui, L2TP (Layer Two TUnneling Protocol) qui est une évolution des deux précédents.

- **Protocole de niveau 3** comme IPSec (Internet Protocol Security), développé par un groupe de travail du même nom à l'IETF [15] depuis 1992, MPLS (MultiProtocol Label Switching) principalement utilisé par les fournisseurs d'accès Internet.

- **Protocole de niveau 4** comme SSL/TLS (Secure Socket Layer/Tranports Layer Security).

4.1.5 Schéma d'un VPN

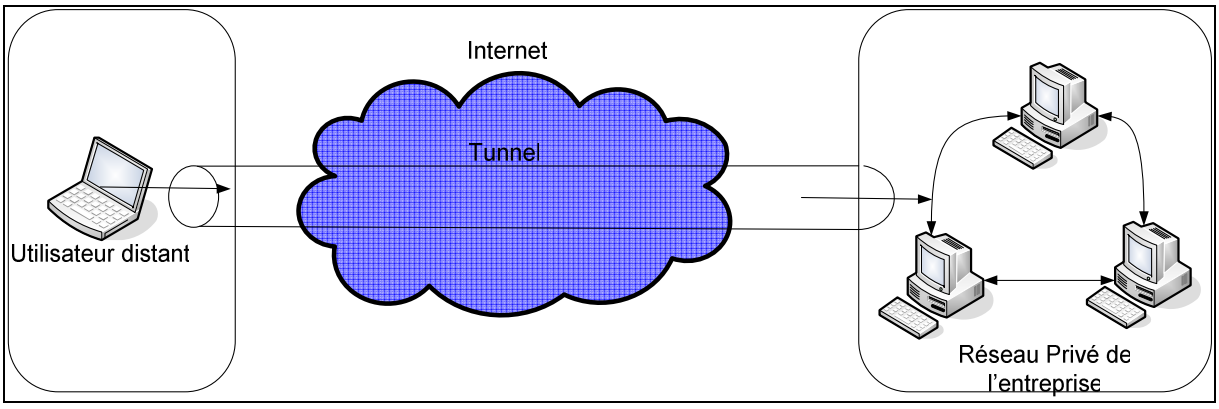


Figure 4-1: Fonctionnement VPN

[15] Internet Engineering Task Force

4.2 Openvpn

4.2.1 Présentation

Comme expliqué plus haut, Openvpn est un logiciel libre de droit et il est donc une très bonne alternative en ce qui concerne la création de tunnels protégés utilisant Internet comme support physique. Openvpn se base sur le protocole de sécurisation SSL/TLS pour assurer la confidentialité des échanges, ce protocole a été élaboré dans le but de protéger les données via des navigateurs web (ex: https [16]). L'authentification se fait la plupart du temps à l'aide de certificats accompagnés de clés.

Il existe deux modes d'utilisations principales concernant Openvpvn:

- **Mode de niveau 2 :** utilisation d'une interface TAP (mode bridgé)
- **Mode de niveau 3 :** utilisation d'une interface TUN (mode routé)

Globalement, si l'on ne souhaite gérer que du trafic IP (niveau 3) il faudra s'orienter vers un mode d'utilisation de niveau 3, tandis que si l'on désire transporter des protocoles autres que IP (niveau 2) il faudra s'orienter vers le mode d'utilisation de niveau 3.

Les paquetages Debian nécessaires à son installation sont les suivants :

- bridge-utils_0.9.6-5_i386.deb -> uniquement pour le serveur (interface TAP)
- openssl_0.9.8a-8_i386.deb
- libssl0.9.8_0.9.8a-8_i386.deb
- liblzo1_1.08-3_i386.deb
- openvpn_2.0.6-1_i386.deb

Tous ces paquetages peuvent être installé à l'aide de la ligne de commande "apt-get install <nom_du_paquet>" sous un environnement Debian.

4.2.2 Création des certificats et des clés RSA

Pour s'authentifier l'un et l'autre, le client et le serveur s'échangent leur certificat et vérifient chacun de leur côté la validité de celui-ci avec le certificat de l'autorité de certification.
Pour générer ces documents, j'ai utilisé un fichier de configuration d'Openssl joint en annexes.

Voici les commandes Unix nécessaires :

- Création d'un certificat autosigné

```
openssl req –nodes –new –x509 –keyout /path/cacert.pem –days 3650 –
config /path_vers_le_fichier_de_conf_openssl
```

- Création de la clé et de la requête de certificat à faire signer par l'autorité de certification

```
openssl req –nodes –new –keyout server.key –out server.csr –config
/path_vers_le_fichier_de_conf_openssl
```

- Vérification et signature de la requête de certificat

```
openssl ca –out server.crt –in server.csr –config
/path_vers_le_fichier_de_conf_openssl
```

Ces deux étapes sont à répéter pour créer un certificat pour le client à signer par la même autorité de certification.

- Création des paramètres Diffie Hellman pour le cryptage du tunnel

```
openssl dhparam –out dh1024.pem 1024
```

4.2.3 Protocole SSL / TLS

4.2.3.1 A propos d'SSL / TLS

C'est le besoin d'éliminer les interceptions sur les réseaux par des personnes non autorisés et par la même occasion d'empêcher la récupération de données personnels des utilisateurs qui a conduit à l'élaboration d'un standard permettant des transmissions sécurisées des données via les navigateurs Web. Netscape Communications est la compagnie qui a conçu le protocole SSL (Secure Socket Layer), qui fut le premier protocole de sécurisation des communications via les navigateurs Web.

La première version du protocole a été publiée dans le milieu de l'année 1994 et est intégrée au navigateur Mosaic, puis fin 1994 la seconde version fut intégrée dans le navigateur Netscape Navigator. La troisième version fut publiée fin 1995 permettant de corriger les faiblesses de son prédécesseur.

C'est en mai 1996 que l'IETF accepta de faire de SSL un standard universel. Et c'est en 1999 que SSL fut renommé par l'IETF : TLS (Transport Layer Security), TLS v1.0 n'est qu'une extension de SSL v3.0.

A l'heure actuelle la plupart des navigateurs Web supportent le protocole et c'est ainsi que le protocole SSL est utilisé dans les transactions sur Internet allant de l'achat de livres jusqu'au transfert de fonds.

4.2.3.2 Généralités

SSL est donc un protocole à négociation développé à l'origine par Netscape. Son but est de sécuriser les transactions Internet, par authentification du serveur et éventuellement du client, et par chiffrement de la session.

Il est une couche optionnelle se situant entre les couches d'application et de transport. Le but de SSL est d'être un protocole de sécurité facile à déployer assurant une sécurité totale des échanges de plusieurs applications.

Pour TCP, SSL n'est qu'une application qui utilise ses services.

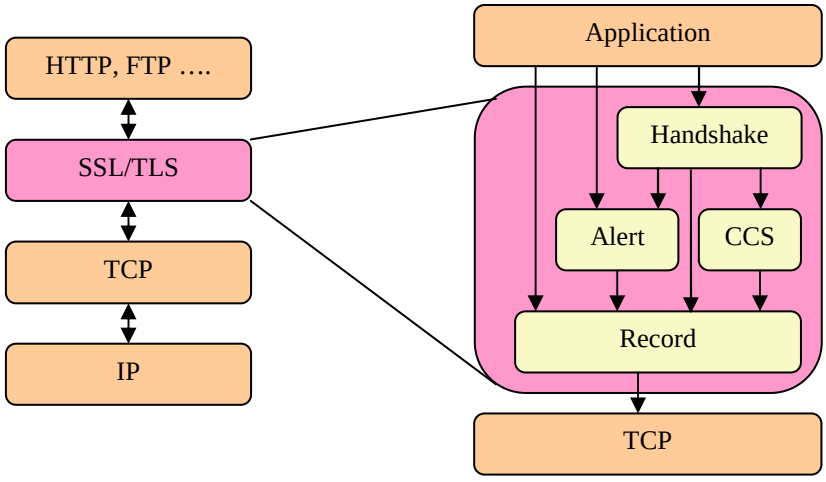


Figure 4-2 : Le protocole SSL/TLS dans son environnement

SSL ne dépend pas des applications utilisées lors des transactions et s'applique sous les protocoles HTTP, FTP, Telnet, LDAP, etc.

[16] HyperText Transfert Protocol Security

La sécurisation des connexions à l'aide du protocole SSL doit assurer que :

- La connexion assure la confidentialité des données transmises
- La connexion assure l'intégrité des données transmises
- L'identité des correspondants peut être vérifiée
- La connexion est fiable

Clients et serveurs commencent par s'authentifier mutuellement (pas toujours de manière symétrique), puis négocient une clé symétrique de session qui servira à assurer la confidentialité des transactions. L'intégrité de ces dernières est assurée par l'application de HMAC [17].

4.2.3.3 Fonctionnalités

Les principales fonctions assurées par SSL sont :

- **Authentification :** dans SSL v3.0 et TLS, l'authentification du serveur est obligatoire. Elle a lieu à l'ouverture de la session. Elle emploie pour cela des certificats conformes à la recommandation X 509 v3. Cela permet au client de s'assurer de l'identité du serveur avant tout échange de données. Dans la version actuelle de SSL et de TLS l'authentification du client reste facultative.

- **Confidentialité :** Elle est assurée par des algorithmes de chiffrement symétriques. Bien que le même algorithme soit utilisé par les deux parties chacune possède sa propre clé secrète qu'elle partage avec l'autre. Les algorithmes utilisés sont : DES, 3DES, RC2, RC4.

- **Intégrité :** Elle est assurée par l'application d'un algorithme de hachage (SHA ou MD5) aux données transmises. L'algorithme génère, à partir des données et d'une clé secrète, une signature pour les données appelée code d'authentification de message. Ainsi tout changement appliqué aux données provoquera un message d'erreur côté récepteur puisque la signature contenue dans le message sera différente de celle calculée par le récepteur.

4.2.3.4 Principe de fonctionnement

Le protocole SSL/TLS est composé de 4 sous protocoles qui sont :

- **Handshake Protocol** : Ce protocole permet l'authentification obligatoire du serveur, du client (optionnelle), et la négociation de la suite du chiffrement qui sera utilisé lors de la session.

- **CCS (ChangeCipherSpec) Protocol** : Ce protocole comprend un seul et unique message (1 octet) qui porte le même nom que le protocole, il permet d'indiquer au protocole Record la mise en place des algorithmes de chiffrement qui viennent d'être négociés.

- **Alert Protocol** : Ce protocole génère des messages d'alerte suite aux erreurs que peuvent s'envoyer le client et le serveur. Les messages sont composés de 20 octets, le premier étant soit *fatal* soit *warning*. Si le niveau de criticité du message est fatal, la connexion SSL est abandonnée.

- **Record Protocol** : Ce protocole intervient après l'émission du message ChangeCipherSpec. Il permet de garantir la confidentialité à l'aide de chiffrement des données et l'intégrité à l'aide de génération d'un condensât.

[17] Hashed Message Authentification Code

Le protocole SSL comporte une phase de négociation où client et serveur peuvent :

- définir le niveau de sécurité voulu
- s'authentifier

Après cette phase, ils peuvent communiquer (échange de données applicatives : HTTP, FTP, etc.)

Rappelons que les trois fonctionnalités principales de SSL sont l'authentification du serveur, l'authentification du client et le chiffrement des données. Le protocole se compose de deux couches principales : le protocole Record qui traite l'encodage des données à envoyer et le protocole Handshake qui gère la négociation.

Nous allons détailler les diverses étapes du processus de négociation, avec les messages que s'envoient le client et le serveur qui permettent la construction de la communication sécurisée grâce à SSL.

4.2.3.5 Processus de négociation client/serveur

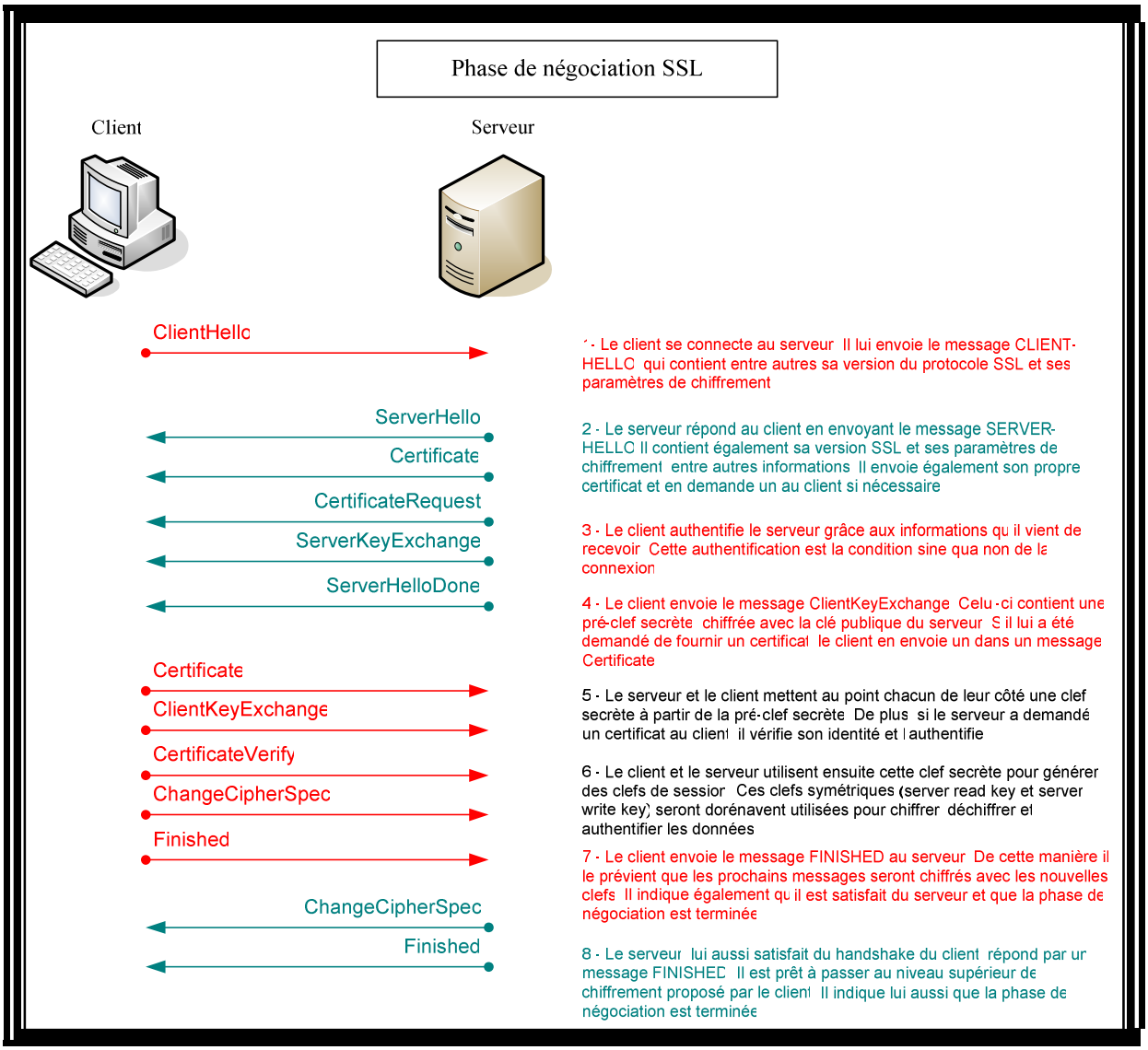


Figure 4-3 : Négociation SSL Client/Serveur

Suite à cette phase de négociation, le client et le serveur peuvent s'échanger des données de façon sécurisée. Celles-ci seront chiffrées par l'algorithme symétrique choisit au cours de la négociation.

4.2.4 Interfaces TUN / TAP

4.2.4.1 Présentation

Les "devices" TUN et TAP permettent de créer une interface réseau virtuelle

- au niveau IP (couche 3); on parlera alors de TUN
- au niveau Ethernet (couche 2); on parlera alors de "device TAP"

et permettent à des processus d'envoyer directement des paquets IP (TUN) ou des trames Ethernet (TAP) sur ces interfaces, via l'utilisation d'un fichier spécial (/dev/net/tun ou /dev/net/tap). Elles permettent également à des processus de recevoir les paquets IP ou les trames Ethernet arrivant sur les interfaces virtuelles.

Les interfaces réseaux peuvent être utilisées exactement comme des interfaces normales (physiques, comme des cartes ethernet, par exemple). On peut leur associer des adresses IP, émettre des paquets/trames dessus, ou faire du routage entre interfaces virtuelles et interfaces physiques (ou entre interfaces virtuelles).

Une des utilisations possibles de ces devices est la création de tunnels en mode utilisateur, c'est-à-dire récupérer le trafic arrivant sur une interface TUN ou TAP, et l'expédier ailleurs, encapsulé dans des messages ou des connexions selon un autre protocole (udp ou tcp, par exemple), et réémettre, sur une interface TUN ou TAP, du trafic reçu encapsulé dans des messages ou des connexions selon un autre protocole. Un traitement intermédiaire (compression, chiffrement) peut être intercalé au moment de la réception ou de l'envoi du trafic sur les devices TUN/TAP. VTun et OpenVPN sont des exemples de tunnels utilisant des devices TUN/TAP.

4.2.4.2 Mode "bridged" : interface TAP (niveau 2 du modèle OSI)

L'interface TAP est une interface de niveau 2 (couche liaison de données) et utilise donc des trames Ethernet.

Lorsque deux hôtes sont dans le même sous réseau (connecté au même switch par exemple), ils communiquent entre eux en utilisant l'adresse MAC du destinataire, c'est ce que l'on appelle une communication de niveau 2.

Dans ce mode de configuration, l'interface de connexion au VPN du serveur doit être une interface bridgées, c'est-à-dire qu'il faut faire un pont entre l'interface virtuelle TAP et une interface physique comme ethX sous Linux (avec X le numéro d'identification de l'interface), cette interface bridgé sera appelée brX (avec X un identifiant de l'interface).

A l'opposé étant donné que le client n'a nul besoin, en principe, de faire suivre les trames Ethernet qu'il reçoit, son interface se limitera à une simple interface virtuelle TAP.

L'utilisation de ce mode est utile dans le cas où l'on souhaite faire communiquer des réseaux au niveau 2 (Ethernet) ; comme par exemple dans le cas où l'on souhaite pouvoir retrouver les mêmes vlans dans deux réseaux reliés par un VPN ou bien transporter un autre protocole de même niveau que IP.

4.2.4.3 Mode "routed" : interface TUN (niveau 3 du modèle OSI)

L'interface TUN est une interface de niveau 3 (couche IP) et utilise comme il se doit des paquets IP. Etant donné que dans ce type de configuration nos deux hôtes, client/serveur ne se trouve pas dans le même sous réseau les paquets devront être routés via une passerelle.

Lorsque deux hôtes ne sont pas dans le même sous réseau la seule solution pour communiquer est d'utiliser une communication dite de niveau 3, c'est-à-dire en utilisant les adresses IP des destinataires. Les paquets IP sont ensuite routés par des routeurs qui peuvent ensuite communiquer en niveau 2 avec les clients (pc ou équipements tels que switchs) du sous réseau.

Comme il s'agit ici d'une interface virtuelle de niveau 3 (IP) il n'est nullement nécessaire d'effectuer un pont entre celle-ci et une interface physique elle aussi de niveau 3. Il suffira éventuellement de modifier la table de routage.

4.2.4.4 Lancement d'Openvpn

- **Mode "bridged" :** La création d'une interface TAP puis d'un pont s'effectue à l'aide de quelques commandes unix simples que l'on peut regrouper dans un unique fichier à exécuter une fois pour toutes avant le lancement du serveur (fournis en annexe).

  - Création d'une interface TAP :

```
openvpn --mktun --dev tap0
```

  - Création du pont :

```
brctl addbr br0
```

  - Ajout des interfaces dans le pont :

```
brctl addif br0 eth0
brctl addif br0 tap0
```

  - Configuration IP des interfaces du pont :

```
ifconfig tap0 0.0.0.0 promisc up
ifconfig eth0 0.0.0.0 promisc up
ifconfig br0 192.168.X.254 netmask 255.255.255.0 broadcast 192.168.X.255 up
```

Remarques:

- Le mode "promisc up" permet de pouvoir voir tout le trafic passant.
- La configuration IP de l'interface br0 est donnée comme exemple avec une adresse de classe C mais n'est pas significative.

  - Lancement du service openvpn :

```
openvpn --config /path_du_fichier_de_config
```

ou

```
openvpn --daemon --config /path_du_fichier_de_config
```

si l'on veut lancer le processus en tant que démon.

- **Mode "routed" :**

Il n'y a aucune configuration préalable à effectuer dans ce mode configuration. Une fois les fichiers de configuration édités de chaque côté du futur tunnel, il suffit de lancer le processus Openvpn de la même manière que pour une configuration en mode "bridged".

4.2.4.5 Avantages et inconvénients

- **TAP**
  - Avantages :
    - Fonctionne avec n'importe quel protocole de niveau trois (IP, IPX…)
    - Ne nécessite pas de routage
  - Inconvénients :
    - Reçois tout le traffic (broadcast)
    - Beaucoup de bande passante gaspillée (entêtes ethernet)
    - Plus lent que TUN

- **TUN**
  - Avantages :
    - Ne reçoit que le traffic lui étant destiné
    - Moins de bande passante utilisée
    - Plus rapide que TAP
  - Inconvénients :
    - Nécessite d'être routé
    - N'autorise que du traffic IP

4.2.5 Test de débit

Pour valider le fait que les débits avec une interface TUN sont plus rapides j'ai procédé à quelques tests de vitesses à l'aide d'un outils sous Debian : iperf

Son utilisation est relativement simple puisqu'il fonctionne sur un modèle client/serveur :

- **Serveur :**

```
iperf –s                        # test de connexion tcp
```

ou

```
iperf –s -u                     # test de connexion udp
```

Une fois ceci fait le serveur est en attente de connexion et écoute sur toutes les interfaces.

- **Client :**

```
iperf –c <@ip serveur>          # test de connexion tcp par défaut 100Mbits/s
```

ou

```
iperf –c <@ip serveur> -u       # test de connexion udp par défaut 1Mbits/s
```

ou

```
iperf –c <@ip serveur> -b 10M   # test de connexion udp à 10Mbits/s
```

ou

```
iperf –c <@ip serveur> -b 100M # test de connexion udp à 100Mbits/s
```

- Les tests ont donné les résultats suivants avec TAP ou TUN, pour différents algorithmes de chiffrement et avec/sans compression :

	TCP							
	Simple	Lzo	BL+Lzo	BL	DES+Lzo	3DES+Lzo	AES128+Lzo	AES256+Lzo
TAP	85	153	126	83	131	119	126	131
TUN	90	159	135	84	136	125	131	133

**test de vitesse en TCPMbits/s*

	UDP [100Mbits/s]							
	Simple	Lzo	BL+Lzo	BL	DES+Lzo	3DES+Lzo	AES128+Lzo	AES256+Lzo
TAP	85	100	100	84	100	95	100	100
TUN	90	100	100	85	100	100	100	100

**test de vitesse en UDP 100Mbits/s*

- **Légendes :**
  - Simple : ni compression, ni chiffrement
  - Lzo : compression uniquement
  - BL : chiffrement Blowfish
  - DES/3DES : chiffrement DES/3DES
  - AES128/256 : chiffrement AES 128 bits ou 256 bits
  - Tous les débits sont donnés en Mbits/s

Les machines utilisées pour ces tests sont un PC de bureau Intel Pentium 4 cadencé à 3GHz avec 256 Mo de mémoire vive et un PC portable Intel Pentium M 1,73GHz avec 1Go de mémoire vive. Il ressort de ce test qu'effectivement l'utilisation d'une interface TUN est plus rapide et principalement lors de connexion TCP.

Remarque importante:

Une observation supplémentaire concernant la consommation de pourcentage CPU lors de ces tests démontre que la compression est très gourmande en ressources, de l'ordre de 55-60% sans et 80% avec celle-ci, et cela malgré le changement d'algorithme de chiffrement qui influe peu sur cette consommation.

Cette remarque devra être tenu en compte lors du choix de la technologie à utiliser pour créer des VPNs, en effet la compression peut grandement ralentir la machine en charge des VPNs et ainsi ralentir les utilisateurs. Si le besoin d'augmentation de bande passante ne se fait pas ressentir il est bien entendu possible d'ignorer cette option.

4.2.6 Architecture retenue

Pour répondre aux besoins du service Réaumur, qui sont notamment la nécessité de traçage de connexion et l'appartenance à un vlan, il a été nécessaire de se pencher davantage sur une configuration précise.

Etant donné qu'il faut pouvoir contrôler les connexions au niveau vlan il est nécessaire d'utiliser des interfaces de niveau 2 (TAP) et dans la configuration utilisée, nous faisons tourner autant d'instances du serveur Openvpn que de VLANs, et donc autant d'interfaces VLANs virtuelles sur le serveur.

4.2.6.1 Schéma d'architecture

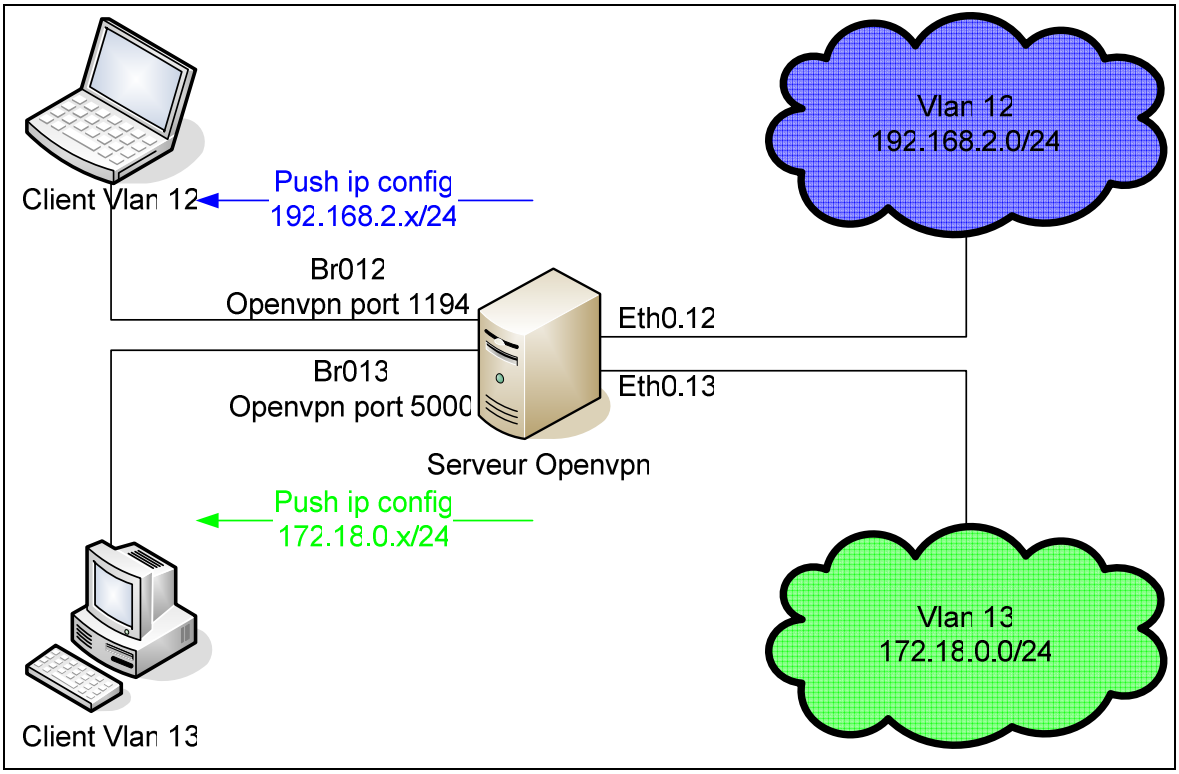


Figure 4-4: Schéma d'architecture Openvpn

4.2.6.2 Explication

Cette architecture met en œuvre autant d'instance du serveur Openvpn que de Vlan auquel il est rattaché, il faut donc jouer sur le numéro de port pour pouvoir gérer différents Vlans.

Il est recommandé d'utiliser Openvpn avec le protocole UDP pour éviter le "TCP over TCP", c'est-à-dire l'encapsulation de TCP par lui-même pouvant entraîner des problèmes de retransmissions beaucoup trop longue dus au mécanisme de fenêtrage.

Dans cet exemple, le client VPN du VLAN 12 devra se connecter sur le port 1194 de l'instance du serveur Openvpn lui correspondant et le client VPN du VLAN 13 devra donc se connecter sur le port 5000 de l'instance du serveur Openvpn correspondant à son VLAN. Cela oblige bien entendu à fournir aux différents clients le numéro de port sur lequel ils devront se connecter.

L'attribution d'une adresse IP se fera en fonction du Common Name présent sur le certificat du client lors de la connexion au serveur ; en effet le serveur doit obligatoirement posséder un fichier, portant le même nom que le Common Name, dans lequel la configuration IP fixe pour ce client est présente. Ainsi même si un client se connecte sur une autre instance du serveur que la sienne, aucune adresse ne lui sera attribuée et il n'aura pas accès au réseau.

4.3 IOS Cisco

4.3.1 Présentation

Dans cette partie nous allons étudier une autre méthode pour créer des VPNs, il s'agit d'utiliser un routeur Cisco, tel qu'un 2851 et de changer son IOS, si besoin, pour un autre plus spécifique et permettant de créer des tunnels IPSec avec un chiffrement 3DES : la version utilisée est la 12.4. L'authentification par nom d'utilisateur et mot de passe sera effectuée ici via un serveur radius opensource Freeradius inclus dans la distribution Debian utilisée.

Il s'agit ici d'utiliser un routeur Cisco en tant que concentrateur VPN sur lequel arriveront toutes les demandes de connexions VPN, les connexions seront établies à l'aide du protocole IPSec [18] et chiffrées à l'aide de l'algorithme de chiffrement 3DES.

4.3.2 Protocole IPSEC

4.3.2.1 A propos d'IPSec

IPsec se présente sous la forme d'un standard, développé par un groupe de travail du même nom à l'IETF depuis 1992. Une première version basique de cette extension d'IP a paru, sous forme de RFC, en 1992. Développé à l'origine dans le cadre de la future version d'IP, à savoir IPv6, IPsec a réussi à se porter à la version actuelle IPv4, en attendant la mise en service à grande échelle d'IPv6, et est maintenant disponible pour tous.

4.3.2.2 Généralités

IPsec est un protocole destiné à fournir différents services de sécurité. Il propose ainsi plusieurs choix et options qui lui permettent de répondre de façon adaptée aux besoins des entreprises, particuliers, etc…
IPsec s'insère dans la pile de protocole TCP/IP, au niveau d'IP. Cela signifie qu'il agit sur chaque paquet IP reçu ou émis et peut soit le laisser passer sans traitement particulier, soit le rejeter, soit lui appliquer un mécanisme de sécurisation.

Ainsi toutes les implémentations d'une pile de protocole TCP/IP d'IPv6 intègre nativement IPsec, en revanche IPsec est optionnel pour la version actuelle d'IPv4 et n'est pas encore fourni en standard sur la plupart des systèmes courants.
Le placement d'IPsec au niveau IP, c'est-à-dire au niveau réseau, présente l'avantage de le rendre exploitable par les niveaux supérieurs, et donc d'offrir un moyen de protection unique pour toutes les applications.

Le réseau IPv4 étant largement déployé et la migration vers IPv6 nécessitant encore beaucoup de temps, il est vite apparu intéressant de définir des mécanismes de sécurité qui soient communs à la fois à IPv4 et IPv6.

[18] Internet Protocol Security

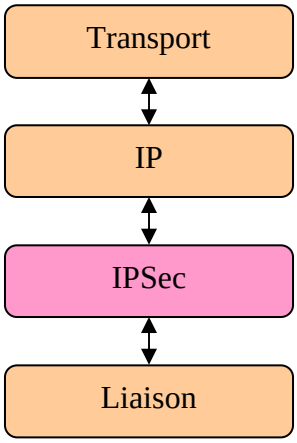


Figure 4-5: IPSec dans le modèle OSI

4.3.2.3 Fonctionnalités

Les principales fonctions que peut assurer le protocole IPsec sont :

- **Authentification des données :** permet de s'assurer, pour chaque paquet échangé, qu'il a bien été émis par la bonne machine et qu'il est bien à destination de la seconde machine.

- **Authentification des extrémités :** Cette authentification mutuelle permet à chacun de s'assurer de l'identité de son interlocuteur à l'établissement du tunnel. Elle s'appuie sur le calcul d'intégrité pour garantir l'adresse IP source.

- **Confidentialité des données :** IPSec permet si on le désire de chiffrer le contenu de chaque paquet IP pour éviter la lecture de ceux-ci par quiconque. Elle est assurée par un chiffrement symétrique des données.

- **Intégrité des données :** IPSec permet de s'assurer qu'aucun paquet n'a subit de modification quelconque durant son trajet en rajoutant à chaque paquet IP le résultat d'un calcul de hachage (SHA-1 ou MD5) portant sur tout ou partie du datagramme.

- **Protection contre les écoutes et analyses de trafic :** IPsec permet de chiffrer les adresses IP réelles de la source et de la destination, ainsi que toute l'en-tête IP correspondant.

- **Protection contre le rejeu :** IPSec permet de se prémunir des attaques consistant à capturer un ou plusieurs paquets dans le but de les envoyer à nouveau pour bénéficier des mêmes avantages que l'envoyeur initial. Elle est assurée par la numérotation des paquets IP et la vérification de la séquence d'arrivée des paquets.

4.3.2.4 Principe de fonctionnement

IPSec fait appel à deux mécanismes de sécurité pour le trafic IP :

- **AH (Authentication header) :** Le protocole AH assure l'intégrité des données en mode non connecté et l'authentification de l'origine des datagrammes IP sans chiffrement des données. Son principe est d'ajouter un bloc au datagramme IP. Une partie de ce bloc servira à l'authentification, tandis qu'une autre partie, contenant un numéro de séquence, assurera la protection contre le rejeu.

- **ESP (Encapsulation Security Payload) :** Le protocole ESP assure, en plus des fonctions réalisées par AH, la confidentialité des données et la protection partielle contre l'analyse du trafic, dans le cas du mode tunnel (voir ci-dessous). C'est pour ces raisons que ce protocole est le plus largement employé.

La technologie IPSec présente deux modes de fonctionnement :

- **Le mode transport** : prend un flux de niveau transport (couche de niveau 4 du modèle OSI) et réalise les mécanismes de signature et de chiffrement puis transmet les données à la couche IP. Dans ce mode, l'insertion de la couche IPsec est transparente entre TCP et IP. TCP envoie ses données vers IPsec comme il les enverrait vers IPv4.
  L'inconvénient de ce mode réside dans le fait que l'en-tête extérieur est produit par la couche IP c'est-à-dire sans masquage d'adresse. De plus, le fait de terminer les traitements par la couche IP ne permet pas de garantir la non utilisation des options IP potentiellement dangereuses. L'intérêt de ce mode réside dans une relative facilitée de mise en œuvre, il est donc utilisé pour créer des tunnels entre des machines.

- **Le mode tunnel** : Dans le mode tunnel, les données envoyées par l'application traversent la pile de protocole jusqu'à la couche IP incluse, puis sont envoyées vers le module IPsec. L'encapsulation IPsec en mode tunnel permet le masquage d'adresses en plus de ce qu'autorise le mode transport. Ce mode est donc utilisé pour créer des tunnels entre des réseaux.

Avant que les paquets puissent être sécurisés par IPSec, une SA (Security Associations ou Associations de Sécurité en français) doit exister. Elle peut être créée manuellement ou dynamiquement. Le protocole IKE [19] est utilisé pour la création dynamique de cette SA; il s'agit d'un protocole hybride basé sur les protocoles ISAKMP [20] (voir ci-dessous), Oakley et SKEME [21]: il utilise les bases de ISAKMP, les modes de Oakley et les techniques de partage des clés de SKEME. Ces trois protocoles étant des protocoles d'échanges de clés.

[19] Internet Key Exchange
[20] Internet Security Association and Key Management Protocol
[21] Secure Key Exchange Mecanism for intErnet

4.3.3 Protocoles ISAKMP/Oakley/SKEME

4.3.3.1 Présentation

Ces trois protocoles d'échange de clé ont pour but d'effectuer l'échange des divers paramètres de chiffrement et d'authentification nécessaire à la future communication. Ces paramètres sont notamment la clé, l'identification des deux parties et trois algorithmes utilisés durant l'authentification :

- Chiffrement : pour assurer la confidentialité des échanges
- Hachage : une fonction pseudo aléatoire pour protéger l'intégrité des messages et l'authentification des champs de ces messages
- Authentification : l'algorithme sur lequel est basé l'authentification mutuelle des deux parties.

4.3.3.2 Protocole ISAKMP

4.3.3.2.1 Rôle

Le protocole ISAKMP, utilisé par IPSec pour la création de tunnel, à pour rôle d'établir, de négocier, de modifier ou de supprimer des Associations de Sécurité et leurs attributs.
Les SA contiennent les paramètres de sécurité suivants :

- Algorithme de chiffrement et taille des clés
- Clé de session
- Choix du protocole AH ou ESP
- Mode tunnel ou transport

4.3.3.2.2 Principe de fonctionnement

ISAKMP se déroule en deux phases :

- Création de la SA ISAKMP, qui servira à la sécurisation de l'ensemble des échanges futurs: on a donc négociation d'attributs relatifs à la sécurité, vérification des identités des tiers, génération des clés...

- Négociation de paramètres de sécurité relatifs à une SA à établir pour un mécanisme donné (par exemple AH ou ESP), via la SA ISAKMP établie en phase 1.

4.3.3.2.3 Echange ISAKMP

Il existe 5 types d'échanges ISAKMP différents :

- **Base Exchange :** l'échange de base permet l'échange simple et rapide de toutes les données nécessaires à la communication, mais ne fournit aucune sécurité supplémentaire (comme la confidentialité...).

- **Identity Protection Exchange :** l'échange de protection d'identité assure l'anonymat des tiers, en n'effectuant leur authentification que sous la protection des systèmes mis en place avec l'échange des données de génération de secret partagé.

- **Authentification Only Exchange :** l'échange d'authentification seul permet uniquement l'authentification des tiers.

- **Aggressive Exchange :** l'échange aggressif assure les mêmes services que l'échange de base, mais en un nombre minimal de messages (il contracte en fait en un seul message les données de négociation de la SA, d'authentification et d'échange de clé); on notera qu'il empêche du même coup l'utilisation de l'échange de clés selon Diffie-Hellman.

- **Informational Exchange :** l'échange d'information est un message univoque d'information concernant la gestion des SA d'un tiers (modification, suppression, ...).

4.3.3.3 Protocole Oakley

Le protocole Oakley est un protocole d'échange de clé basé sur Diffie-Hellman (DH = chiffrement à clé publique).
Voici quelques caractéristiques de ce protocole :

- **Cookies :**
  - précaution contre les attaques de type «déni de service» en les rendant plus difficile.
  - identifiant unique basé sur les informations de connexions (un peu comme un identifiant de socket)
  - utilisé à chaque message durant les communications

- **Groupes prédéfinis :**
  - Paramètres globaux DH fixés
  - DH et ECDH (Elliptic Curve DH) réguliers

- **Unicitée :**
  - Evite les attaques de type "replay attack" ou attaque de type "rejeu" en français

- **Authentification :**
  - Via des systèmes de chiffrements symétrique ou asymétrique

- **Trois modes de fonctionnement :**
  - **Main Mode Exchange** permettant d'utiliser la propriété de Perfect Forward Secrecy (PFS). Cette propriété signifie que la découverte des secrets à long terme (clé privée RSA) ne compromet pas la clé session, elle n'est pas présente lors d'un échange de clé de session simplement chiffrée à l'aide d'une clé publique. Ce mode permet aussi d'assurer l'anonymat des deux parties.
  - **Aggressive Mode Exchange**
  - **Quick Mode Exchange** (décrit ci-dessous)

4.3.3.4 Protocole SKEME

SKEME est un protocole d'échange de clés proposant plusieurs modes de fonctionnement. Le mode de fonctionnement basic est basé sur l'utilisation de clés publiques et la génération d'un secret partagé Diffie-Hellman. Cependant SKEME n'est pas restreint à l'utilisation de clés publiques, mais autorise aussi l'utilisation d'une clé pré-partagée. Cette clé peut être obtenue par distribution manuelle ou par l'intermédiaire d'un centre de distribution de clés tel que Kerberos. Le centre de distribution des clés permet la communication d'un secret entre deux machines via un troisième intermédiaire de confiance. L'utilisation de ce secret pour l'authentification du secret Diffie-Hellman et pas directement en tant que clé de session diminue la nécessité de faire confiance au centre de distribution des clés. SKEME permet aussi d'échanger les clés plus rapidement en supprimant l'utilisation de Diffie-Hellman lorsque la propiété de PFS n'est pas requise.

SKEME contient quatres modes disctincts:

- Le mode basic qui fournit un échange de clés basé sur une clé publique et assure la propriété de PFS grâce à Diffie-Hellman
- Un échange de clés basé sur l'utilisation d'une clé publique, mais sans Diffie-Hellman
- Un échange de clés basé sur l'utilisation d'une clé pré-partagée et Diffie-Hellman
- Un mécanisme de saisie de nouvelles clés basé simplement sur un algorithme de chiffrement symétrique.

De plus SKEME est composé de trois phases : SHARE, EXCH et AUTH.

- Durant la phase **SHARE**, les deux protagonistes s'échangent des demi-clés, chiffrées respectivement avec leur clé publique. Ces deux demi clés sont ensuite utilisées pour créer une clé secrète. Si l'anonymat est requis, les identités des deux extrémités sont aussi chiffrées. Bien sûr si un secret partagé existe déjà cette phase est passée.
- La phase d'échange **EXCH** est utilisée, en fonction du mode choisi, pour échanger l'une ou l'autre des valeurs publiques Diffie-Hellman ou d'unicité. Le secret partagé DH ne sera créé qu'après la fin de cet échange.
- Les valeurs publiques ou d'unicité sont authentifiées durant la phase d'authentification **AUTH** en utilisant la clé secrète établie durant la phase SHARE.

Les messages de ces trois phases ne doivent pas nécessairement suivre l'ordre décrit ci-dessus, dans la pratique ils sont combinés pour minimiser le nombre des messages d'échanges.
Une autre phase, connue sous le nome de COOKIES, peut être ajouter avant la phase SHARE pour fournir une protection contre les attaques de type "déni de service" grâce au mécanisme de cookies.

4.3.4 Protocole IKE

4.3.4.1 Principe de fonctionnement

IKE est le protocole de gestion des clés implémenté par IPSec. Il comprend 3 modes, qui gèrent les échanges de paramètres entre les entités souhaitant communiquer; le but est de créer la SA dans les deux pairs à l'aide des deux phases d'ISAKMP. La première phase est utilisée pour créer une SA IKE - via les échanges identity protect exchange et aggressive exchange d'ISAKMP -, la deuxième pour négocier les paramètres nécessaires à la création de la SA IPSec.

4.3.4.2 Echange IKE

Le protocole IKE utilise, quant à lui, trois modes différents pour ses échanges :

- **Main Mode Exchange :** Pour l'établissement de la SA IKE, six messages sont utilisés : trois requêtes et trois réponses.
  Cet échange se déroule en trois étapes:
  - échange des paramètres Diffie-Hellman,
  - échange d'aléas,
  - authentification des parties

Le premier échange sert à la négociation des paramètres nécessaires à la mise en place de la SA IKE. Le deuxième échange sert à la négociation des valeurs publiques de l'algorithme Diffie-Hellman et des valeurs pseudo-aléatoires contenues dans le bloc d'aléas (Nonce Payload). Lors du dernier échange les deux pairs s'envoient leurs identités respectives et le bloc Hash nécessaire à l'authentification.

- **Aggressive Mode Exchange :** Ce mode utilise directement l'Aggressive Exchange de ISAKMP; l'échange se déroule donc en seulement trois messages.

- **Quick Mode Exchange :** Une fois la SA IKE établie avec le Main Mode ou l'Aggressive Mode, le Quick Mode est utilisé pour établir une SA pour un autre protocole de sécurité, comme AH ou ESP, sous la protection de la SA IKE précédemment établie. Dans un échange en Quick Mode, les deux pairs négocient les caractéristiques de la SA IPSec à établir, et génèrent les clés correspondantes. La SA IKE protège ces échanges en chiffrant et en authentifiant les messages transmis.

  En plus de l'en-tête ISAKMP, du Hash, de la SA, du Nonce et des paramètres optionnels de Diffie-Hellman, les deux pairs peuvent s'échanger des informations concernant leur identité, comme leur adresse IP.

  La connexion sécurisée ayant été établie par les protocoles ci-dessus, il est alors nécessaire de protéger les données utiles: c'est le rôle des protocoles AH et ESP.

4.3.5 VRF (Virtual Routing and Forwarding)

4.3.5.1 Principe de fonctionnement

La mise en oeuvre de VRFs (ou RAV pour Routage et Acheminement Virtuel, en français) permet de créer plusieurs routeurs logiques au sein d'un même et unique routeur physique, celui-ci étant découpé et chacune des VRFs possédant alors sa propre table de routage qui est totalement indépendante des autres tables de routages des différentes VRFs. On se retrouve alors avec plusieurs routeurs virtuels étanches entre eux.

Cette technologie permet de faire cohabiter des populations diverses sur un même routeur sans nécessairement mettre en oeuvre d'ACLs, ainsi on pourra router des paquets provenant des sites divers et cela de manière sécurisée, c'est à dire sans que les sites ne puissent se voir entre eux et sans avoir besoin de déployer un routeur physique par site. Cette technologie permet également de connecter plusieurs sites distants possédants des plages d'adresses IP pouvant se chevaucher puisque celles-ci seront virtuellement sur des routeurs différents.

Dans notre cas d'utilisation, les VRFs vont nous permettrent d'isoler les différents groupes d'utilisateurs/clients VPN se connectant sur le concentrateur VPN, ainsi on est sûr que chaque utilisateur se retrouvera dans, et uniquement dans, son VLAN de travail d'origine sans pouvoir accéder aux réseaux de travail adjacents auxquels il n'aurait pas le droit d'accéder en tant normal en étant connecté directement sur le réseau.

4.3.5.2 Schéma de fonctionnement

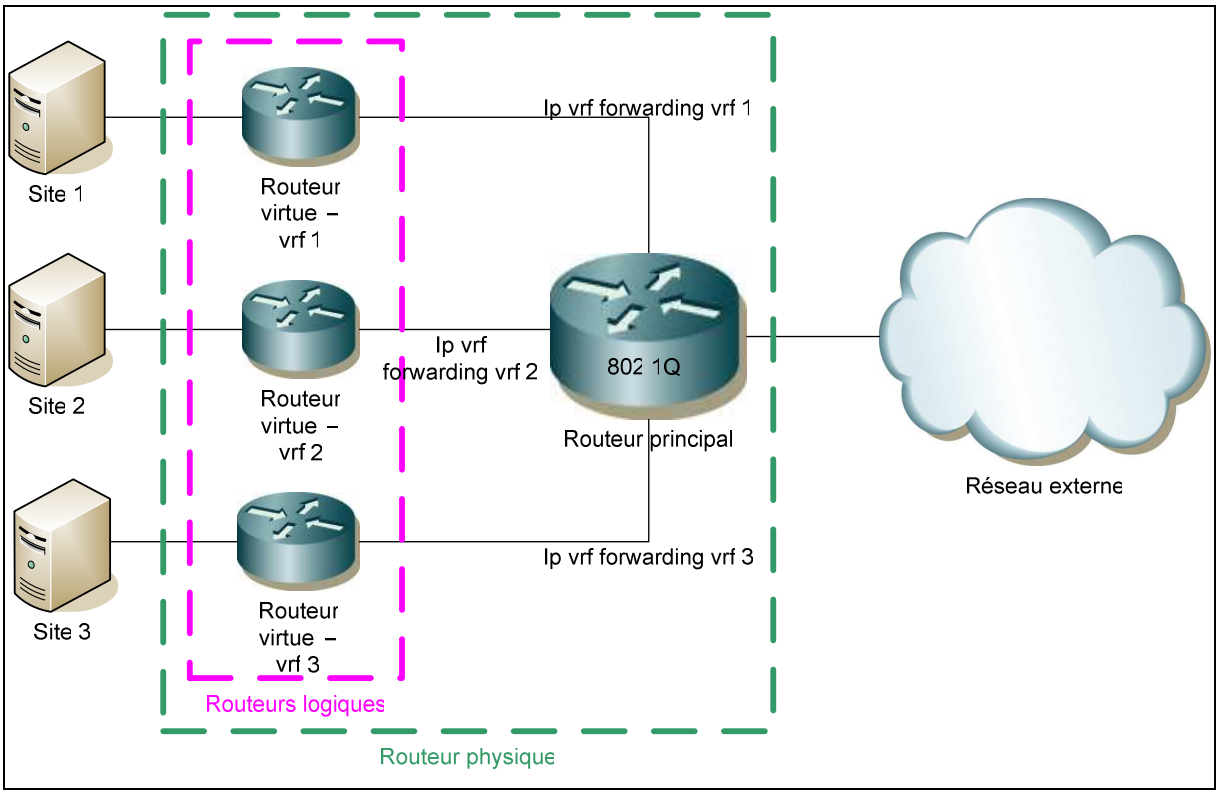


Figure 4-6: Fonctionnement VRFs

La connexion vers les différents sites est matérialisée par la configuration de l'interface interne en sous interfaces écoutant sur les VLANs respectifs aux sites, il s'agit ici du protocole 802.1Q, ainsi chacune de ces sous interfaces appartient à un seul VLAN à écouter, et il est possible de récupérer les trames arrivant de VLANs différents. L'ajout d'un paramètre de configuration, relatif aux VRFs, sur chacune de ces sous interfaces permettant de préciser quelle table de routage appliquer à ces sous interface évitant ainsi la communication intersites. (La configuration du routeur est fournie en annexe).

4.3.6 Authentification du client VPN pour l'établissement du tunnel

4.3.6.1 Principe de fonctionnement

La configuration du modèle de routeur Cisco testé s'est avérée compliquée à mettre en œuvre : elle est basée sur des profils qui sont créés dans le but de définir les systèmes d'authentifications, de chiffrement, de comptabilité ainsi que les VRFs associées. Trois méthodes d'authentifications des clients VPN sont proposés sur le routeur Cisco : secret pré-partagé, clé RSA, certificats. Nous verrons ici l'authentification par clé pré-partagé qui bien qu'étant la plus facile à mettre en œuvre n'est pas dépourvu de faille contrairement à un système basé sur les certificats.

Voilà comment se déroule la configuration du routeur :

- Pour être en mesure de traiter correctement les demandes de connexions arrivant sur une interface Ethernet, il est nécessaire d'appliquer une "crypto map" à une interface Ethernet, cette "crypto map" regroupe tous les paramètres nécessaires au traitement des données.

- Cette "crypto map" pointe vers une "crypto map dynamic", la différence entre les deux étant qu'une map dynamic permet de gérer des connexions venant d'hôtes inconnus.

- Définition de "crypto map dynamic" définissant le "transform-set" et pointant vers un ou plusieurs profils.
- Définition d'un "transform-set" permettant de spécifier les paramètres de chiffrement et de hachage
- Définition d'un profil dans lequel sont précisés :
  - Les paramètres d'authentification, d'autorisation et de comptabilité (AAA via RADIUS ici)
  - La VRF dans laquelle sont définies les routes spécifiques à un réseau
  - Le paramètre "match" permettant de pointer sur un groupe défini ci-dessous (ou sur un certificat)
  - Divers paramètres (paramètres de connexion, description du profil…)
- Définition d'un groupe et d'une clé associée pour l'authentification du client VPN sur le concentrateur (ISAKMP phase 1)

4.3.6.2 Schéma de fonctionnement

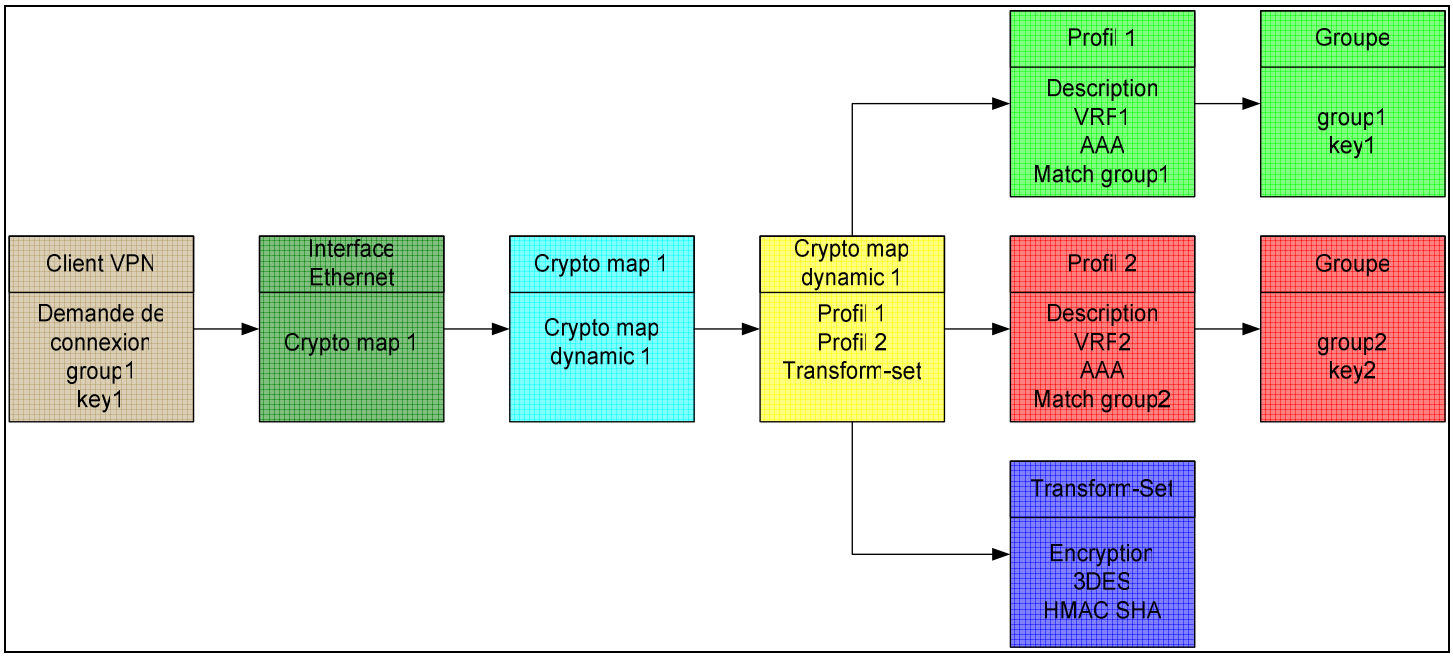


Figure 4-7: Schéma d'authentification

4.3.7 Authentification de l'utilisateur

4.3.7.1 Présentation

Dans la configuration utilisée, nous allons authentifier les clients VPNs en fonction de leur nom d'utilisateur et de leur mot de passe enregistrés sur un serveur de type RADIUS [22]. Dans le cas précis sur lequel nous travaillons, le client VPN se connecte au routeur/concentrateur VPN et celui-ci va aller vérifier l'authentification cliente auprès du serveur RADIUS distant. Ce dernier retournera alors la configuration IP correspondante au client nouvellement enregistré, puis le concentrateur VPN relayera cette configuration au client pour que celui-ci puisse avoir accès au réseau depuis son VLAN d'origine.

Le serveur RADIUS utilisé est Freeradius qui est, comme expliqué plus haut, un serveur open source, c'est-à-dire que les codes sources de ce logiciel sont libre de droit et publié sur Internet. Ce logiciel est de plus inclus en tant que paquetage dans toutes les distributions basées sur Debian. Le paquetage nécessaire à son utilisation est "freeradius" dans sa version actuelle 1.1.0-1.1 installable via la ligne de commande "apt-get install freeradius" sous un environnement Debian.

De nos jours la sécurité est devenue primordiale est s'axe autour de trois grands points, à savoir, les AAA pour Authentication-Authorization-Accounting ou en français Authentification-Autorisation-Comptabilité.

4.3.7.2 AAA

- **Authentication :** Cela correspond à l'identification fiable de l'utilisateur, que ce soit une personne physique ou un service. Cette identification passe par la présentation de l'identité de l'utilisateur. Cette information est unique à chaque utilisateur et non sécrète. Elle sert de référence dans la base des utilisateurs. Le contrôle de cette information consiste à vérifier un secret partagé entre l'utilisateur et le serveur. Elle peut être de plusieurs types :
  - **Statique :** l'information transmise est alors la même lors d'authentifications successives : mot de passe type UNIX par exemple.
  - **Dynamique :** on passe alors par un challenge entre le serveur et l'utilisateur, ce qui permet d'avoir une information différente à chaque nouvelle authentification : calculette, carte à puce…
  - **Biométrique :** reconnaissance vocale, empreintes, iris

- **Authorization :** C'est le fait de déterminer quels sont les droits de l'utilisateur. Par exemple, après s'être loggé, l'utilisateur peut essayer certaines commandes. L'autorisation détermine si l'utilisateur peut ou non les utiliser. Dans certaines implémentations, l'identification et l'autorisation sont regroupées en une seule étape.

- **Accounting :** Cela consiste à mesurer les ressources qu'un utilisateur consomme, en terme d'échange réseau, de ressources système… Cela sert en fait à journaliser un certain nombre d'informations sur l'utilisateur. Cela permet de connaître à la fois les services demandés par l'utilisateur et la quantité de ressources requises. Ces informations peuvent être souvent utilisées dans des buts de facturation.

[22] Remote Access Dial-In User Service

4.3.8 Protocole RADIUS

4.3.8.1 Présentation

Le protocole RADIUS a été créé par Livingston et normalisé par l'IETF sous la forme des RFCs 2138 et 2139. Le fonctionnement de RADIUS est basé sur un système client/serveur chargé de définir les accès d'utilisateurs distants à un réseau. Il s'agit du protocole de prédilection des fournisseurs d'accès à internet car il est relativement standard et propose des fonctionnalités de comptabilité permettant aux FAI de facturer précisément leurs clients.

Le protocole RADIUS repose principalement sur un serveur (le serveur RADIUS), relié à une base d'identification (fichier local, base de données, annuaire LDAP, etc.) et un client RADIUS, appelé NAS (Network Access Server), faisant office d'intermédiaire entre l'utilisateur final et le serveur. Le mot de passe servant à authentifier les transactions entre le client RADIUS et le serveur RADIUS est chiffré et authentifié grâce à un secret partagé.

Il est à noter que le serveur RADIUS peut faire office de proxy, c'est-à-dire transmettre les requêtes du client à d'autres serveurs RADIUS.

Protocole	UDP port 1812 : authentification/autorisation Port 1813 : comptabilité
Chiffrement	Chiffrement du mot de passe à l'aide du secret partagé
Architecture	Autorisations liées à l'authentification
Emission du profile	Profile global envoyé au NAS [23] à la fin de l'authentification
Protocoles non supportés	ARA [24] et NetBEUI [25]
Challenge/réponse	Unidirectionnelle

4.3.8.2 Principe de fonctionnement

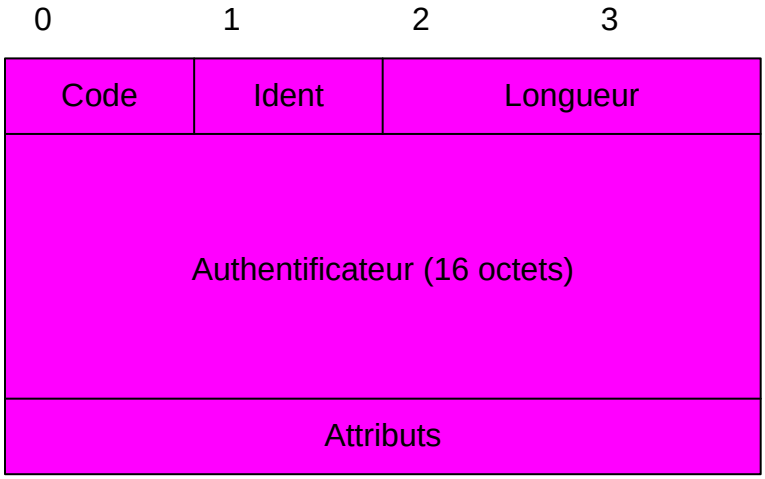
Le fonctionnement de RADIUS est basé sur un scénario proche de celui-ci :

- Un utilisateur envoie une requête au NAS afin d'autoriser une connexion à distance,
- Le NAS achemine la demande au serveur RADIUS,
- Le serveur RADIUS consulte la base de données d'identification afin de connaître le type de scénario d'identification demandé pour l'utilisateur. Soit le scénario actuel convient, soit une autre méthode d'identification est demandée à l'utilisateur. Le serveur RADIUS retourne ainsi une des quatre réponses suivantes pour ce qui est de l'authentification :
  - **ACCEPT :** l'identification a réussi ;
  - **REJECT :** l'identification a échoué ;
  - **CHALLENGE :** le serveur RADIUS souhaite des informations supplémentaires de la part de l'utilisateur et propose un « défi » (en anglais « challenge ») ;
  - **CHANGE PASSWORD :** le serveur RADIUS demande à l'utilisateur un nouveau mot de passe.

Suite à cette phase dite d'authentification, débute une phase d'autorisation où le serveur retourne les autorisations de l'utilisateur. Dans le cas des implémentations de RADIUS, ces deux étapes d'authentification et d'autorisation sont regroupées en une seule phase.

[23] Network Access Server
[24] Apple Remote Access
[25] NetBIOS Enhanced User Interface (NetBIOS = Network Basic Input Output System)

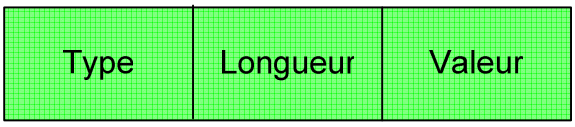
4.3.8.3 Format des trames RADIUS



- **Code :** Identifie le type du message

1 Access-Request
2 Access-Accept
3 Access-Reject
4 Accounting-Request
5 Accounting-Response
11 Access-Challenge
12 Status-Server (expérimental)
13 Status-Client (expérimental)
255 Reserved

- **Identificateur :** associe requêtes et réponses (1 octet)
- **Longueur :** longueur de toutes les zones données (2 octets)
- **Authentificateur :** utilisé pour authentifier la réponse du serveur et pour protéger les mots de passe (16 octets)
- **Attributs :** format (type, longueur, valeur) utilisé pour véhiculer toutes les informations nécessaires



Les valeurs des attributs sont les suivantes :

1 User-Name	2 User-Password	3 CHAP-Password
4 NAS-IP-Address	5 NAS-Port	6 Service-Type
7 Framed-Protocol	8 Framed-IP-Addres	9 Framed-IP-Netmask
10 Framed-Routing	11 Filter-Id	12 Framed-MTU
13 Framed-Compression	14 Login-IP-Host	15 Login-Service
16 Login-TCP-Port	17 (unassigned)	18 Reply-Message
19 Callback-Number	20 Callback-Id	21 (unassigned)
22 Framed-Route	23 Framed-IPX-Network	24 State
25 Class	26 Vendor-Specific	

4.3.8.4 Schéma de fonctionnement

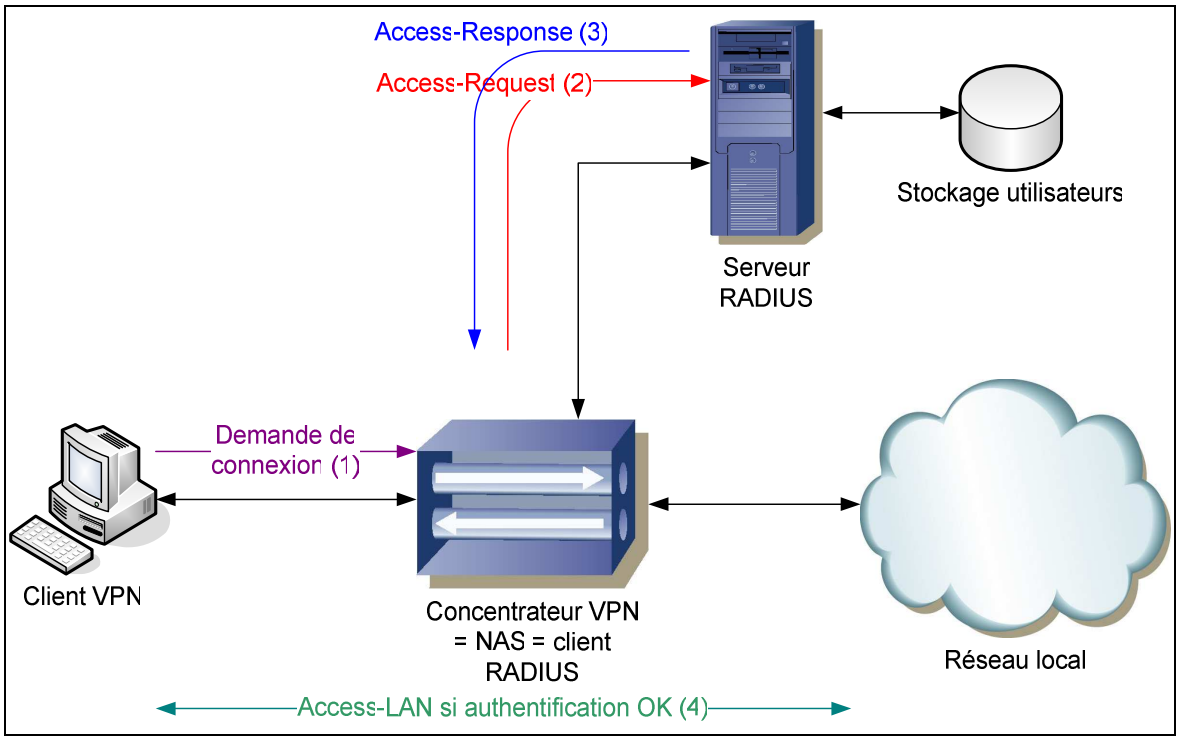


Figure 4-8: Fonctionnement RADIUS

4.3.8.5 Mise en œuvre

Pour l'utilisation de la maquette, le serveur RADIUS va ici nous permettre d'attribuer une adresse IP fixe en fonction de l'utilisateur qui vient de se connecter, ainsi on sera toujours sûr de l'identité d'un individu connecté sur le réseau. L'attribution d'adresse IP permet aussi d'envoyer une configuration, au client, en fonction de son VLAN d'origine et de la VRF appliquée à l'interface concernée ; cette configuration permet d'assurer qu'aucun changement de réseau n'est possible par l'utilisateur.

Trois fichiers de configuration sont essentiels pour le bon fonctionnement du serveur (fournis en annexes) :

- **Radiusd.conf :** configuration général (port, répertoire, module utilisés, comptabilité…)
- **Clients.conf :** configuration des clients (NAS) autorisés à se connecter au serveur RADIUS
- **Users :** configuration des profils des utilisateurs (login, mot de passe, configuration IP…)

5 Conclusions

5.1 Conclusion générale

5.1.1 DHCP

Le paramétrage d'un serveur DHCP à l'aide de ce que l'on peut appeler "DHCP évolué" permet beaucoup plus qu'une simple attribution de configuration réseau à un client en faisant la demande. Il est possible d'effectuer cette attribution en fonction de nombreux paramètres tels que le lieu de connexion de l'utilisateur ou encore son VLAN d'appartenance.

Il est cependant à noter que cette présentation n'a pas la prétention de fournir une configuration type et une liste exhaustive de paramètres et de directives pouvant être utilisés, cette configuration s'applique ici à un environnement bien défini qu'est l'attribution d'adresse IP fixe en fonction du lieu de connexion. Il faut également savoir que ce modèle de configuration s'applique non seulement dans ce cas précis mais également avec des logiciels adaptés et avec la version de DHCP cité ci-dessus, aucune autre version n'ayant été testé, il en va de même en ce qui concerne le commutateur utilisé pour les tests.

Ce nouveau service rendu aux utilisateurs, sera mis en production prochainement et permettra une meilleure utilisation du réseau avec une configuration grandement simplifié. On limite ainsi le nombre de demandes d'aide ou de dépannage émanant des utilisateurs novices principalement, et on accroît l'autonomie des populations utilisatrices du réseau de campus.

5.1.2 VPNs

Nous avons pu voir ici deux types de connexions client/serveur VPN, la première utilisant un logiciel libre et un serveur de type PC, et la seconde faisant appel à du matériel Cisco encore une fois joint à un serveur de type RADIUS. Et c'est cette dernière qui sera retenu et éventuellement améliorée pour parer à des problèmes de sécurité connus.

Les problèmes pouvant apparaître sur cette deuxième configuration touche l'authentification par mot de passe puisque celui-ci peut éventuellement être intercepté sur le système client par faute de chiffrement solide et efficace. Toutefois ce genre de problème peut être résolu en utilisant une authentification par certificat sans mot de passe pré-partagé et qui reste cependant plus complexe à mettre en oeuvre.

Il est également à noter que cette configuration a été testé sur 3 modèles de routeurs différents que sont les Cisco 2620, 1811 et 2851. Un problème de compatibilité a été trouvé concernant le premier de ces modèles (2620), puisque bien qu'aucun bugs ne soit répertoriés entre ce routeur et l'image utilisé et supportant les VPN/IPSec (image cisco 12.3(18)), la configuration refusait de fonctionner et un simple changement de routeur vers des modèles plus récents (1811 et 2851) embarquant déjà des IOS supportant les fonctionnalités requises, s'est avéré salutaire.

Malgré tout, ce problème de compatibilité m'a permit de fouiller dans les documentations de configuration Cisco et découvrir de nombreuses options paramétrables, ainsi que plusieurs modèles de routeur Cisco.

Ces deux services s'inscrivent dans la mission du service Réaumur qui est de fournir toujours davantage de services aux utilisateurs dans le but de leur simplifier l'utilisation du réseau universitaire, ainsi de nouveaux accès sécurisés à distance et une configuration IP dynamique simple d'utilisation viennent enrichir la liste des nombreuses options proposées par le service de gestion du réseau aquitain.

5.2 Conclusion personnelle

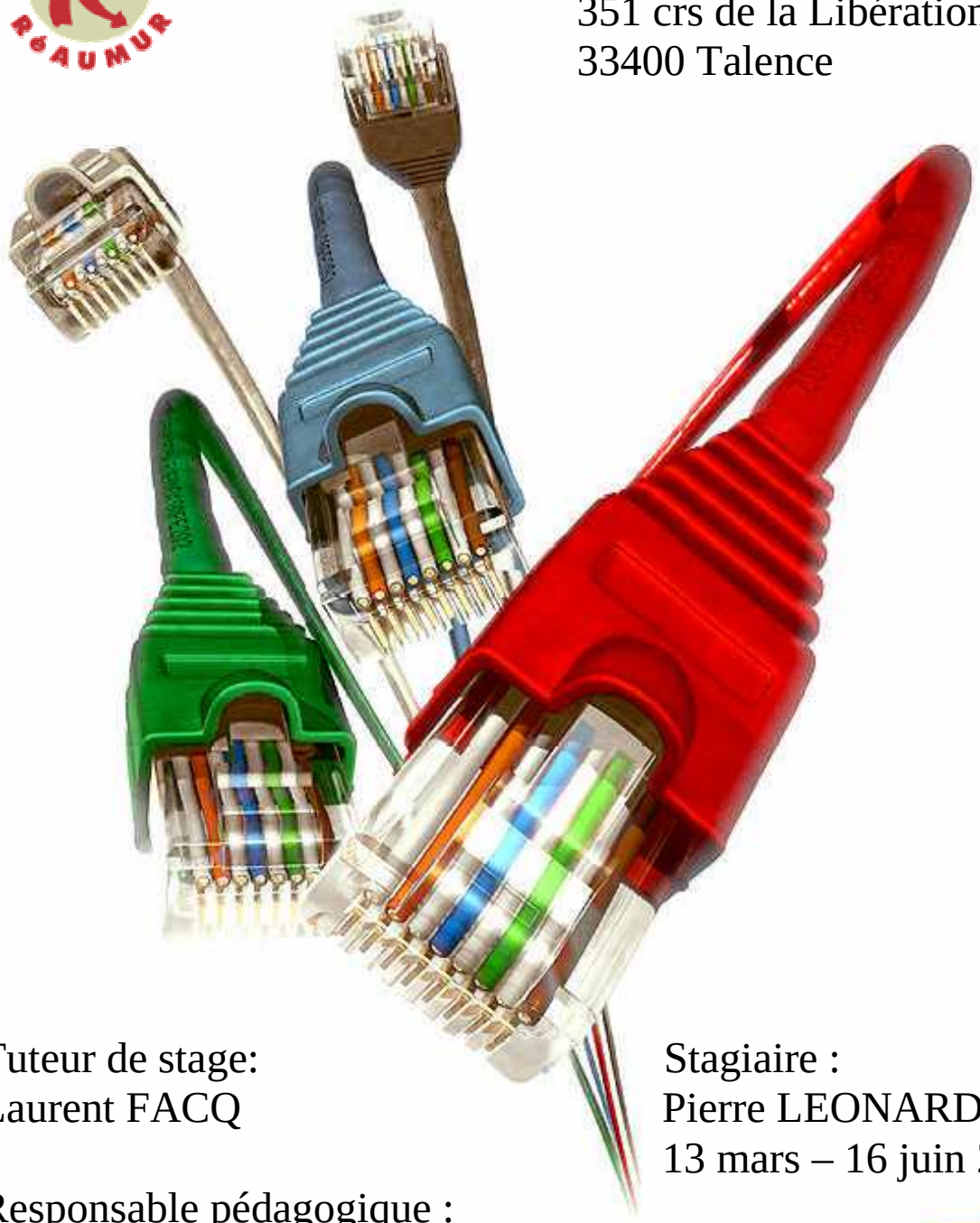
Ce stage aura été vraiment enrichissant pour moi, j'ai pu travailler sur du matériel relativement chère et possédant de nombreuses fonctionnalités, cela m'a également permis de découvrir de nombreuses choses concernant les réseaux informatiques. Je me suis heurté à beaucoup de problèmes et c'est en réfléchissant dessus pendant de long moment que j'ai pu comprendre en profondeur le fonctionnement d'un serveur DHCP et des VPN Openvpn et Cisco. Si j'ai appris une chose importante pendant ce stage c'est que l'on n'apprend rien lorsque tout marche de suite sans avoir à se creuser la tête.

Ces trois mois m'auront donc permit d'utiliser beaucoup de matériel (switch et routeur) mais également apprendre un nouveau langage de programmation/scripting qu'est le PERL, car il s'agit là encore d'une découverte dont je suis à peu près sûre qu'elle me resservira dans ma futur vie professionnelle d'administration et de sécurité des réseaux informatiques.

Au-delà de tout cela les contacts humains socioprofessionnels m'ont beaucoup apporté dans ma culture générale informatique et me permettent ainsi d'être plus solide et intéressé sur de nombreux sujets ayant trait à l'informatique en général. Je suis donc très fier d'avoir pu effectuer ce stage dans un environnement serein et sain avec toute l'aide dont je pouvais avoir besoin et cela à tout moment ; c'est la raison pour laquelle je tiens à remercier une nouvelle fois mon tuteur de stage Laurent FACQ mais également tout le personnel du service Réaumur qui a su l'épauler lorsque celui était occupé.

Mobilité et Sécurité
Annexes

Université Bordeaux 1
Service Réaumur
351 crs de la Libération
33400 Talence



Tuteur de stage:
Laurent FACQ

Stagiaire :
Pierre LEONARD
13 mars – 16 juin 2006

Responsable pédagogique :
Christophe BAILLOT

1 Solution DHCP

1.1 Script Perl

Dhcp_gen_conf.pl

```
#!/bin/perl

print "Génération des fichiers de configuration dhcp...\n";
# ouverture du fichier de conf utilisateur
open (CONF,'conf_utilisateur') || die "Pb d'ouverture de
\'conf_utilisateur\' : $!";

# lecture du fichier de conf utilisateur
while (<CONF>) {
      if ($_ =~ /^\s/) {$_ =~ s/^\s*//;} # suppression des blancs en début
de lignes

      if ($_ =~ /^#/ ) {next;}      # ne pas tenir compte des commentaires

      # récupération du chemin d'enregistrement
      if ($_ =~ /ROOT/) {
      @root = split('\s+',$_);
      $root = $root[1];
      $rootclasses = $root."classes.conf";
      if ((@root != 2 and $_ =~ /ROOT/) or $_ !~ /ROOT/) {print STDERR
"chemin d'enregistrement manquant\n";exit;}
      }

      # récupération du nom du réseau
      if ($_ =~ /NETWORK\s+\w/) {
            @network = split('\s+',$_);
            $network = $network[1];
            open (SUBNET,">$root$network.conf") || die "Pb d'ouverture
network: $!";
            print "ecriture fichier $root$network.conf\n";

      # signale un nom network manquant
      if ((@network != 2 and $_ =~ /NETWORK/)) {print STDERR "nom de réseau
manquant\n";}
      }

      # récupération @mac switch
      if ($_ =~ /SWITCH/) {
            $_ =~ s/\s+$//;   # supprime les blancs en fin de ligne
            @switch = split('\s+',$_,4);
            $switch_name = $switch[1];
            # ouverture du fichier de conf switch
            open (SWITCH,">>$root$switch_name.conf") || die "Pb d'ouverture
de \'$switch_name.conf\': $!";
            print "ecriture fichier $root$switch_name.conf\n";
            $mac = $switch[2];
            if (@switch == 3) {$type = cisco;}
            else {$type = $switch[3];}
      }

      # récupération du vlan
      if ($_ =~ /VLAN/) {@vlan = split('\s+',$_);}
      $vlan = $vlan[1];
      # conversion en hexa
      $vlanHexa = sprintf "%04x",$vlan[1];
```

```
	$vlanHexa = substr($vlanHexa,0,2).':'.substr($vlanHexa,2,2);

	# récupération de : module / port / @ip
	if ($_ =~ /^PORT\s+\d+\s+\d+\s/ and $network[1] !~ /\s/) {
		$_ =~ s/\n$//;
		@port = split('\s+',$_,5);
		if (@port == 5){$description = $port[4];}

		$module = $port[1];
		$moduleHexa = sprintf "%02x",$module;	# conversion hexa du
module

		if ($type eq "cisco") {
			$port = sprintf "%02x",$port[2]-1;  # conversion hexa du
port
		}else {$port = sprintf "%02x",$port[2];}

		$ip = $port[3];

		$classes_name = $network."-vlan ".$vlanHexa."-mod
".$moduleHexa."-port ".$port." on ".$switch_name;
		# ouverture du fichier de conf des classes
		open (CLASSES,">>$rootclasses") || die "Pb d'ouverture de
\'classes.conf\' : $!";
		print "ecriture fichier ".$rootclasses."\n";

		# écriture dans le fichier de conf des classes
		print CLASSES "class \"".$classes_name."\"\{\n\tmatch if
\(substring \(option agent\.circuit-id,2, 2\) = ".$vlanHexa."\) and
\(substring \(option agent\.circuit-id,4,1\) = ".$moduleHexa."\) and
\(suffix \(option agent\.circuit-id, 1\) = ".$port."\) and \(suffix\(option
agent\.remote-id,6\) = ".$mac."\)\;\n\}\n";
		# fermeture du fichier de conf des classes
		close (CLASSES);

		# écriture dans le fichier de conf des pools
		print SUBNET "pool \{\n\tallow members of
\"".$classes_name."\";\n\trange ".$ip.";\n\}\n\n";

		$interface = $port[2];

		# écriture dans les fichiers de conf des switchs
		print SWITCH "interface FastEthernet $type $module/$interface\n
description: $description\n ip access-group IP:$ip in\n switchport access
vlan $vlan\n switchport mode access\n access-list IP:$ip permit 0.0.0.0\n
access-list IP:$ip permit $ip\n\n";
	}
}

# fermeture de tous les fichiers encore ouverts
close (CONF);
close (SUBNET);
close (SWITCH);
print "Fichiers générés\n";
```

1.2 Exemple de fichiers de configuration

1.2.1 Fichier utilisateur

Conf_utilisateur

```
ROOT /home/toto/fichiers_generes/

NETWORK name1
SWITCH robert 00:11:22:33:44:55
VLAN 10
PORT 00 20 192.168.0.20 chambre 20
PORT 01 3 192.168.0.3

NETWORK name2
SWITCH albert aa:bb:cc:dd:ee:ff allied-telesyn
VLAN 14
PORT 00 21 172.18.0.2 studio 2
```

1.2.2 Fichier de définitions des classes

Classes.conf

```
class "name1-vlan 00:0a-mod 00-port 13 on robert"{
      match if (substring (option agent.circuit-id,2, 2) = 00:0a) and
(substring (option agent.circuit-id,4,1) = 00) and (suffix (option
agent.circuit-id, 1) = 13) and (suffix(option agent.remote-id,6) =
00:11:22:33:44:55);
}
class "name1-vlan 00:0a-mod 01-port 02 on robert"{
      match if (substring (option agent.circuit-id,2, 2) = 00:0a) and
(substring (option agent.circuit-id,4,1) = 01) and (suffix (option
agent.circuit-id, 1) = 02) and (suffix(option agent.remote-id,6) =
00:11:22:33:44:55);
}
class "name2-vlan 00:0e-mod 00-port 15 on albert"{
      match if (substring (option agent.circuit-id,2, 2) = 00:0e) and
(substring (option agent.circuit-id,4,1) = 00) and (suffix (option
agent.circuit-id, 1) = 15) and (suffix(option agent.remote-id,6) =
aa:bb:cc:dd:ee:ff);
}
```

1.2.3 Fichiers de définitions des pools

Name1.conf

```
pool {
      allow members of "name1-vlan 00:0a-mod 00-port 13 on robert";
      range 192.168.0.20;
}

pool {
      allow members of "name1-vlan 00:0a-mod 01-port 02 on robert";
      range 192.168.0.3;
}
```

Name2.conf

```
pool {
      allow members of "name2-vlan 00:0e-mod 00-port 15 on albert";
      range 172.18.0.2;
}
```

1.2.4 Fichiers de configuration des switchs

Robert.conf

```
interface FastEthernet cisco 00/20
 description: chambre 20
 ip access-group IP:192.168.0.20 in
 switchport access vlan 10
 switchport mode access
 access-list IP:192.168.0.20 permit 0.0.0.0
 access-list IP:192.168.0.20 permit 192.168.0.20

interface FastEthernet cisco 01/3
 description:
 ip access-group IP:192.168.0.3 in
 switchport access vlan 10
 switchport mode access
 access-list IP:192.168.0.3 permit 0.0.0.0
 access-list IP:192.168.0.3 permit 192.168.0.3
```

Albert.conf

```
interface FastEthernet allied-telesyn 00/21
 description: studio 2
 ip access-group IP:172.18.0.2 in
 switchport access vlan 14
 switchport mode access
 access-list IP:172.18.0.2 permit 0.0.0.0
 access-list IP:172.18.0.2 permit 172.18.0.2
```

1.2.5 Fichier de configuration dhcp

Dhcpd.conf

```
include "/home/toto/fichiers_generes/classes.conf";

# option definitions common to all supported networks
option domain-name "test.u-bordeaux.fr";

default-lease-time 600;
max-lease-time 7200;

# Use this to send dhcp log messages to a different log file (you also
# have to hack syslog.conf to complete the redirection).
log-facility local7;

#Display tag in syslog when an user is connected
if exists agent.circuit-id
{
      log ( info, concat( "Lease for ", binary-to-ascii (10, 8, ".",
leased-address), " is connected to interface ",binary-to-ascii (10, 8, "/",
suffix ( option agent.circuit-id, 2)), " (add 1 to port number!), VLAN ",
binary-to-ascii (10, 16, "", substring( option agent.circuit-id, 2, 2)),  "
on switch ", binary-to-ascii(16, 8, ":", substring( option agent.remote-id,
2, 6))));
}

shared-network test {
            subnet 192.168.0.0 netmask 255.255.255.0 {
            authoritative;
            include "/home/toto/fichiers_generes/name1.conf";
            }

            subnet 172.18.0.0 netmask 255.255.255.0 {
            authoritative;
            include "/home/toto/fichiers_generes/name2.conf";
            }
}
```

2 Solutions VPN

2.1 Openvpn

2.1.1 Fichier de configuration d'Openssl

Openssl-vpn.cnf

```
HOME            = .
RANDFILE        = $ENV::HOME/.rnd

# Extra OBJECT IDENTIFIER info:
oid_section        = new_oids

[ new_oids ]

####################################################################
[ ca ]
default_ca    = CA_default

####################################################################
[ CA_default ]

dir         = /etc/ssl/openvpn
certs         = $dir/certs
crl_dir         = $dir/crl
database    = $dir/index.txt
#unique_subject    = no

new_certs_dir    = $dir/newcerts

certificate    = $dir/cacert.pem
serial        = $dir/serial

crl         = $dir/crl.pem
private_key    = $dir/private/cakey.pem
RANDFILE    = $dir/private/.rand

x509_extensions    = usr_cert

name_opt     = ca_default
cert_opt     = ca_default

default_days    = 365
default_crl_days= 30
default_md    = md5
preserve    = no

policy        = policy_match

[ policy_match ]
countryName         = match
stateOrProvinceName    = match
organizationName    = match
organizationalUnitName    = optional
commonName        = supplied
emailAddress        = optional
```

```
[ policy_anything ]
countryName         = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName      = optional
commonName          = supplied
emailAddress            = optional

####################################################################
[ req ]
default_bits            = 1024
default_keyfile         = privkey.pem
distinguished_name      = req_distinguished_name
attributes          = req_attributes
x509_extensions     = v3_ca
string_mask = nombstr

[ req_distinguished_name ]
countryName             = Country Name (2 letter code)
countryName_default             = FR
countryName_min                 = 2
countryName_max                 = 2

stateOrProvinceName             = State or Province Name (full name)
stateOrProvinceName_default     = Gironde

localityName                = Locality Name (eg, city)
localityName_default            = bordeaux

0.organizationName          = Organization Name (eg, company)
0.organizationName_default      = pierre

organizationalUnitName          = Organizational Unit Name (eg, section)
organizationalUnitName_default      = test reaumur

commonName              = Common Name (eg, YOUR name)
commonName_max                  = 64

emailAddress                = Email Address
emailAddress_max            = 64

[ req_attributes ]
challengePassword           = A challenge password
challengePassword_min           = 4
challengePassword_max           = 20

unstructuredName            = An optional company name

[ usr_cert ]

basicConstraints=CA:FALSE

nsComment               = "OpenSSL Generated Certificate"



subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always

[ v3_req ]
```

```
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment

[ v3_ca ]

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always,issuer:always

basicConstraints = CA:true

[ crl_ext ]

authorityKeyIdentifier=keyid:always,issuer:always
```

2.1.2 Fichiers de configuration d'Openvpn

2.1.2.1 Mode Bridged – interface TAP (configuration retenue)

Bridge-start

```
#!/bin/bash

openvpn --mktun --dev tap0          # création des interfaces virtuelles
openvpn –-mktun –-dev tap1          # tap0 et tap1

brctl addbr br012                   # création des interfaces de bridge
brctl addbr br013                   # br012 et br013

vconfig add eth0 12                 # création des interfaces vlan
vconfig add eth0 13                 # virtuelles eth0.12 et eth0.13

brctl addif br0 eth0.12             # ajout de l'interface eth0.12 au bridge
brctl addif br0 tap0                # ajout de l'interface tap0 au bridge

brctl addif br0 eth0.13             # ajout de l'interface eth0.13 au bridge
brctl addif br0 tap1                # ajout de l'interface tap1 au bridge

ifconfig tap0 0.0.0.0 promisc up    # configuration de l'interface tap0
ifconfig eth0.12 0.0.0.0 promisc up # configuration de l'interface eth0.12
ifconfig tap1 0.0.0.0 promisc up    # configuration de l'interface tap1
ifconfig eth0.13 0.0.0.0 promisc up # configuration de l'interface eth0.13
ifconfig eth0 0.0.0.0 promisc up    # configuration de l'interface eth0

# configuration des interfaces de pont br012 et br013
ifconfig br012 192.168.2.254 netmask 255.255.255.0 broadcast 192.168.2.255
ifconfig br013 172.18.0.254 netmask 255.255.255.0 broadcast 172.18.0.255
```

Bridge-stop

```
#!/bin/bash
```

```
ifconfig br012 down                # désactivation des bridges
ifconfig br013 down

brctl delbr br012                  # suppression des bridges
brctl delbr br013

ifconfig tap0 0.0.0.0              # remise à zéro de la configuration tap0
ifconfig tap1 0.0.0.0              et tap1

vconfig rem eth0.12                # suppression des interfaces vlan
vconfig rem eth0.13                virtuelles

# reconfiguration initiale de l'interface eth0
ifconfig eth0 192.168.0.1 netmask 255.255.255.0 broadcast 192.168.0.255
```

Configuration Serveur

```
local <@ip>             # ip publique du serveur
port 1194               # port à changer pour deuxième serveur
proto udp

dev tap0                # type interface = tap (layer2) (à changer pour
tun-mtu 1500            deuxième serveur)
mssfix

persist-key
persist-tun
ca /etc/openvpn/tls/cacert.pem          # certificat de l'autorité de
                                        certification
cert /etc/openvpn/tls/vpn.pierre.crt    # certificat du serveur
key /etc/openvpn/tls/vpn.pierre.key     # clé privé du serveur
dh /etc/openvpn/tls/dh1024.pem          # clé DH pour initialisation du
                                        tunnel
tls-auth /etc/openvpn/tls/ta.key        # clé partagée

server-bridge 192.168.2.254 255.255.255.0 192.168.2.5 192.168.2.15
chroot /etc/openvpn/jail
client-config-dir ccd       # dossier contenant les associations CN/IP
ccd-exclusive               # fichiers d'associations obligatoires
client-to-client            # autorise les clients à se voir entre eux

keepalive 10 120
cipher BF-CBC               # chiffrement Blowfish
comp-lzo                    # compression des données => meilleures perfs

max-clients 15
user nobody
group nogroup

status /etc/openvpn/jail/log/status_bridged.log
log-append /etc/openvpn/jail/log/openvpn_bridged.log

verb 4
mute 10
```

Configuration Client

```
client
```

```
dev tap                         # type interface tap (layer2)
proto udp
remote <@ip> 1194               # ip publique du serveur à joindre + port

resolv-retry infinite
nobind

persist-key
persist-tun
ca /etc/openvpn/tls/cacert.pem           # certificat de l'autorité de
                                         certification
cert /etc/openvpn/tls/nomade1.pierre.crt # certificat du client
key /etc/openvpn/tls/nomade1.pierre.key  # clé privée du client
tls-auth /etc/openvpn/tls/ta.key         # clé partagée

keepalive 10 60
cipher BF-CBC                   # chiffrement Blowfish
comp-lzo                        # compression des données => meilleures perfs

chroot /etc/openvpn/jail/log
log-append /etc/openvpn/jail/log/openvpn_bridged_client_tap.log
verb 2
mute 5
```

2.1.2.2 Mode routed – interface TUN

Configuration Serveur

```
local <@ip>                     # ip publique du serveur
port 1194
proto udp

dev tun                         # type interface = tun (layer3)
tun-mtu 1500
mssfix

persist-key
persist-tun
ca /etc/openvpn/tls/cacert.pem           # certificat de l'autorité de
                                         certification
cert /etc/openvpn/tls/vpn.pierre.crt     # certificat du serveur
key /etc/openvpn/tls/vpn.pierre.key      # clé privé du serveur
dh /etc/openvpn/tls/dh1024.pem           # clé DH pour initialisation du
                                         tunnel
duplicate-cn

server 192.168.1.0 255.255.255.0

chroot /etc/openvpn/jail
client-config-dir ccd           # dossier contenant les fichiers de conf
                                clients
ccd-exclusive                   # fichier de conf obligatoire sinon client
                                refusé
client-to-client                # autorise les clients à se voir entre eux
keepalive 10 120
cipher BF-CBC                   # chiffrement Blowfish
comp-lzo                        # compression des données => meilleures perfs
max-clients 15
user nobody
group nogroup
```

```
status /etc/openvpn/jail/log/status_routed.log
log-append /etc/openvpn/jail/log/openvpn_routed.log
verb 4
mute 10
```

Configuration Client

```
client
dev tun                 # type interface tun (layer3)
proto udp
remote <@ip> 1194       # ip publique du serveur à joindre + port

resolv-retry infinite
nobind

persist-key
persist-tun
ca /etc/openvpn/tls/cacert.pem           # certificat de l'autorité de
                                         certification
cert /etc/openvpn/tls/nomade1.pierre.crt # certificat du client
key /etc/openvpn/tls/nomade1.pierre.key  # clé privée du client

keepalive 10 60
cipher BF-CBC           # chiffrement Blowfish
comp-lzo                # compression des données => meilleures perfs

chroot /etc/openvpn/jail/log
log-append /etc/openvpn/jail/log/openvpn_routed_client_tun.log
verb 2
mute 5
```

2.2 IOS Cisco

2.2.1 Configuration du routeur Cisco 2851

Configuration routeur

```
Current configuration : 4147 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname vpn2851
!
boot-start-marker
boot-end-marker
!
!
aaa new-model           # activation AAA
!
!
aaa group server radius group-radius     # definition d'un serveur RADIUS
 server 192.168.1.3 auth-port 1812 acct-port 1813
!
# définition du modèle AAA sur le serveur RADIUS
aaa authentication login authentication-radius group group-radius
aaa authorization network authorization-radius local
aaa accounting network accounting-radius start-stop group group-radius
!
aaa session-id common
!
resource policy
!
!
!
ip cef
!
!
ip vrf vrflabo          # définition des VRFs
 rd 1:1
!
ip vrf vrflabo2
 rd 2:2
!
no ip domain lookup
ip domain name u-bordeaux.fr
!
!
voice-card 0
 no dspfarm
!
!
username cisco privilege 15 secret 5 xxxxxxxxxxxxxxxxxxxxx
!
!
no crypto xauth GigabitEthernet0/0
!
crypto isakmp policy 3       # définition des paramètres de sécurité de
 encr 3des                   # la phase 1 d'ISAKMP
 authentication pre-share
```

```
 group 2
!
!
# définition des groupes
crypto isakmp client configuration group test-cisco-vpn
 key key-vpn                  # clé partagée concentrateur VPN/client
 domain u-bordeaux.fr
!
crypto isakmp client configuration group test-toto
 key key-toto
 domain toto.fr
!
# définition des profils
crypto isakmp profile profile-pierre
   description profile test reaumur
   vrf vrflabo                              # définition de la vrf
   match identity group test-cisco-vpn   # identification du groupe
   client authentication list authentication-radius # paramètres AAA
   isakmp authorization list authorization-radius
   client configuration address respond
   accounting accounting-radius
   keepalive 30 retry 5
   initiate mode aggressive              # mode de négociation
crypto isakmp profile profile-toto
   description profile test reaumur toto
   vrf vrflabo2
   match identity group test-toto
   client authentication list authentication-radius
   isakmp authorization list authorization-radius
   client configuration address respond
   accounting accounting-radius
   keepalive 30 retry 5
   initiate mode aggressive
!
!
# définition des paramètres de sécurité utilisé pour chiffrer les données
crypto ipsec transform-set set-cisco-vpn esp-3des esp-sha-hmac
!
# définition des maps dynamiques
crypto dynamic-map map-4 5
 set transform-set set-cisco-vpn   # application des parametres de sécurité
 set isakmp-profile profile-pierre # application du profil
 reverse-route                     # préciser le chemin retour des paquets
crypto dynamic-map map-4 10
 set transform-set set-cisco-vpn
 set isakmp-profile profile-toto
 reverse-route
!
!
# définition de la map principale
crypto map map-5 10 ipsec-isakmp dynamic map-4
!
# paramétrage des interfaces Ethernet
interface GigabitEthernet0/0
 ip address 192.168.1.1 255.255.255.0
 duplex auto
 speed auto
 crypto map map-5
!
!
!
interface GigabitEthernet0/1
```

```
 no ip address
 duplex auto
 speed auto
!
interface GigabitEthernet0/1.1      # sous interface 0/1.1
 encapsulation dot1Q 12             # appartient au VLAN 12
 ip vrf forwarding vrflabo          # faire suivre les trames de la vrflabo
 ip address xxx.xxx.xxx.251 255.255.255.224
 no snmp trap link-status
!
interface GigabitEthernet0/1.2
 encapsulation dot1Q 13
 ip vrf forwarding vrflabo2
 ip address xxx.xxx.xxx.222 255.255.255.224
 no snmp trap link-status
!
!
# table de routage standard & VRFs
ip route 0.0.0.0 0.0.0.0 xxx.xxx.xxx.254
ip route vrf vrflabo xxx.xxx.xxx.224 255.255.255.224 xxx.xxx.xxx.251
ip route vrf vrflabo2 xxx.xxx.xxx.192 255.255.255.224 xxx.xxx.xxx.222
!
!
ip http server
no ip http secure-server
!
!
# définition de la clé partagée NAS/RADIUS
radius-server host 192.168.1.3 auth-port 1812 acct-port 1813 key key-radius
!
control-plane
!
line con 0
line aux 0
line vty 0 4
 privilege level 15
 transport input ssh
!
scheduler allocate 20000 1000
!
webvpn context Default_context
 ssl authenticate verify all
 !
 no inservice
!
!
end
```

2.2.2 Configuration du serveur RADIUS (Freeradius)

Radiusd.conf

```
prefix = /usr
exec_prefix = /usr
sysconfdir = /etc
localstatedir = /var
sbindir = ${exec_prefix}/sbin
logdir = /var/log/freeradius
raddbdir = /etc/freeradius
radacctdir = ${logdir}/radacct
confdir = ${raddbdir}
run_dir = ${localstatedir}/run/freeradius
log_file = ${logdir}/radius.log
libdir = /usr/lib/freeradius
pidfile = ${run_dir}/freeradius.pid

user = freerad
group = freerad

max_request_time = 30
delete_blocked_requests = no
cleanup_delay = 5
max_requests = 1024
bind_address = *
port = 0
hostname_lookups = no
allow_core_dumps = no
regular_expressions     = yes
extended_expressions    = yes
log_stripped_names = no
log_auth = no
log_auth_badpass = no
log_auth_goodpass = no
usercollide = no
lower_user = no
lower_pass = no
nospace_user = no
nospace_pass = no
checkrad = ${sbindir}/checkrad

security {
      max_attributes = 200
      reject_delay = 1
      status_server = no
}

proxy_requests  = yes
$INCLUDE  ${confdir}/proxy.conf
$INCLUDE  ${confdir}/clients.conf
snmp  = no
$INCLUDE  ${confdir}/snmp.conf

thread pool {
      max_servers = 32
      min_spare_servers = 3
      max_spare_servers = 10
      max_requests_per_server = 0
}

# Les modules permettent de définir les traitements à appliquer
```

```
modules {
      preprocess {
            huntgroups = ${confdir}/huntgroups
            hints = ${confdir}/hints
            with_ascend_hack = no
            ascend_channels_per_line = 23
            with_ntdomain_hack = no
            with_specialix_jetstream_hack = no
            with_cisco_vsa_hack = no
      }

      detail {
            detailfile = ${radacctdir}/%{Client-IP-Address}/detail-%Y-%m-%d
            detailperm = 0600
      }

      files {
            usersfile = ${confdir}/users
            acctusersfile = ${confdir}/acct_users
            preproxy_usersfile = ${confdir}/preproxy_users
            compat = no
      }
}

# Les modules sont appliqués aux étapes du traitement d'une requête
authorize {
      preprocess
      files
}

authenticate {
}

preacct {
      preprocess
      files
}

accounting {
      detail
}

session {
}

post-auth {
}

pre-proxy {
}

post-proxy {
}
```

Clients.conf

```
client 192.168.1.1 {
      secret            = key-radius       # clé partagée serveur/NAS
      shortname         = router
      nastype           = other
}
```

Users

```
pierre      User-Password == "pierre"
      Framed-IP-Address = xxx.xxx.xxx.236, # IP à attribuer à “pierre”
      Framed-IP-Netmask = 255.255.255.224

toto  User-Password == "toto"               # IP à attribuer à “toto”
      Framed-IP-Address = xxx.xxx.xxx.193,
      Framed-IP-Netmask = 255.255.255.224
```