Évolution du système VPN de l'entreprise Netapsys

Stéphane Soubeyrand

Conservatoire National des Arts et Métiers
Centre Régional Rhône-Alpes
Centre d'Enseignement de Lyon

MEMOIRE

Présenté en vue d'obtenir
Le DIPLÔME D'INGÉNIEUR CNAM
Spécialité : INFORMATIQUE

Par

STEPHANE SOUBEYRAND

ÉVOLUTION DU SYSTÈME VPN DE L'ENTREPRISE NETAPSYS

Soutenu le 25 mai 2016

Jury

Président :	M. Christophe PICOULEAU	CNAM Paris
Membres :	M. Bertrand DAVID	CNAM Lyon
	M. Daniel MULLER	CNAM Lyon
	M. Sergio SIMONE	Responsable Infrastructures Netapsys
	M. Joyce LAMBERT	Administrateur systèmes et réseau Netapsys

Remerciements

À ma femme et à mon fils, ainsi qu'à toute ma famille, pour leur soutien chaleureux et constant durant toutes ces années d'études au CNAM,

À mon tuteur d'entreprise, M. Sergio Simone, ainsi qu'à mon tuteur CNAM : M. Bertrand DAVID, qui m'ont encadré et soutenu tout au long de cette mission,

À mon collègue, M. Joyce Lambert, qui s'est rendu régulièrement disponible pour m'aider durant toute la durée du projet,

À mon employeur, NETAPSYS, qui m'a permis de réaliser cette mission d'ingénieur,

À mes collègues, sans qui la réalisation de ce projet n'aurait pu aboutir,

À vous, professeurs, personnels administratifs notamment Éléonore GONDEAU, et auditeurs du CNAM que j'ai rencontré tout au long de ces années d'études,

À toutes les personnes qui ont participé, de près ou de loin, à la rédaction et à la correction de ce document,

À tous, j'adresse mes plus chaleureux remerciements et je dédie ce mémoire.

Liste des abréviations

AES
Advanced Encryption Standard 41
BASH
Bourne-Again Shell 43
BGP
Border Gateway Protocol 26
EIGRP
Enhanced Interior Gateway Routing Protocol 27
ESN
Entreprise de Service du Numérique, anciennement SSII 8
GRE
Generic Routing Encapsulation 32
IETF
Internet Engineering Task Force 22
IPsec
Internet Protocol Security 22
OSI
Open System Interconnecion 22
OSPF
Open Shortest Path First 27
RIP
Routing Information Protocol 26
SDN
Software-Defined Networking 58
SHA
Secure Hash Algorithm 41
SPOF
Single Point Of Failure 14
SSL
Secure Sockets Layer 21
VM
Machine Virtuelle 10

Glossaire des termes techniques

Cloud computing : Exploitation de la puissance de calcul ou de stockage de serveurs informatiques distants par l'intermédiaire d'un réseau, généralement Internet.

Chiffrement : procédé de cryptographie grâce auquel on souhaite rendre la compréhension d'un document impossible à toute personne qui n'a pas la clé de (dé)chiffrement.

GNU (projet) : Système d'exploitation dont chaque brique du projet est un logiciel libre utilisable de par sa nature dans des projets tiers en vue de la réalisation d'un système d'exploitation complet et entièrement libre.

Hyperviseur : Plateforme mettant à disposition des machines virtuelles.

Linux : Système d'exploitation associant des éléments essentiels du projet GNU et d'un noyau Linux. Dans le langage courant on trouve souvent l'emploi du terme « Linux » seul pour désigner une distribution du système d'exploitation GNU/Linux.

Machine virtuelle : Logiciel installé dans un ordinateur permettant de simuler le fonctionnement d'un dispositif matériel.

Open Source : Logiciels dont la licence respecte des critères précisément établis par *l'Open Source Initiative*, c'est-à-dire les possibilités de libre redistribution, d'accès au code source et de création de travaux dérivés. Mis à la disposition du grand public, ce code source est généralement le résultat d'une collaboration entre programmeurs.

Pare-feu : Équipement réseau informatique chargé de protéger les réseaux auxquels il est relié tout en autorisant le trafic légitime.

Routeur : Équipement réseau informatique chargé de distribuer les données circulant sur le réseau au bon destinataire. Pour effectuer ces opérations, il utilise une table de correspondance associant ces ports réseau aux sous-réseaux connectés, cette table se nomme table de routage.

VPN : De l'anglais *Virtual Private Network*. Système permettant de créer un lien direct entre des ordinateurs distants.

Table des matières

Remerciements 2
Liste des abréviations 3
Glossaire des termes techniques 4
Table des matières 5
Introduction 7
Chapitre 1 Présentation de l'entreprise et cadre de la mission 8
1.1 Le groupe Netapsys 8
1.2 L'équipe « infrastructure » 9
1.3 La cible recherchée : Infrastructure as Code (IAC) 10
Chapitre 2 Le projet « évolution du système VPN » 13
2.1 Cadre du projet 13
2.2 Missions et objectifs 15
2.3 Budget 16
2.4 Risques et contraintes 17
2.5 Planning 17
Chapitre 3 État de l'art des technologies employées 20
3.1 Réseau Privé Virtuel (VPN) 20
3.2 Le routage dynamique 26
3.3 Les tunnels GRE 32
3.4 Technologies retenues 33
Chapitre 4 Réalisation technique 35
4.1 Maitrise de la technologie 35
4.2 Prérequis et paramétrage initial 37
4.3 Mise en place de plateformes de tests 40
4.4 Industrialisation des configurations 43
4.5 Routage dynamique et pare-feu 47
4.6 Difficultés rencontrées 49
Chapitre 5 Déploiement du nouveau système 50
5.1 Prérequis au déploiement 50
5.2 Réalisations pratiques 51
5.3 Résumé des problèmes techniques rencontrés 53

5.4 Transfert de compétences 54
Chapitre 6 Bilan et perspectives 57
6.1 Perspectives du projet 57
6.2 État du projet 59
6.3 Bilan personnel 59
Conclusion 61
Bibliographies 62
Table des annexes 63
Table des figures 72
Liste des tableaux 73

Introduction

La rédaction de ce mémoire marque la dernière étape d'une formation au Conservatoire Nationale des Arts et Métiers de Lyon depuis près de 5 ans. Le but de cette formation, et de l'obtention du diplôme d'ingénieur, est de détenir une clé supplémentaire me permettant d'ouvrir des portes professionnelles sur des postes à responsabilités.

L'objectif de ce mémoire est quant à lui de démontrer mes capacités à être ingénieur en architecture informatique par l'analyse d'une problématique propre à mon Entreprise : l'évolution du système VPN, en traduisant ce problème en termes techniques et en proposant, après avoir effectué des choix de conception et d'optimisation liées aux contraintes de l'entreprise, une solution et une mise en œuvre adéquate.

Netapsys est une entreprise spécialisée en ingénierie informatique implantée en France et à Madagascar. Les 9 sites qui la composent sont reliés entre eux par un mécanisme de chiffrement qui assure la confidentialité des données. Ce mécanisme étant de moins en moins adapté au besoin de l'entreprise, il a été décidé de le remplacer. C'est l'objet de ce projet débuté en mai 2015. De plus, une nouvelle fonctionnalité qui se nomme le routage dynamique a été ajoutée.

La rédaction de ce mémoire est intervenue avant la fin de la phase de déploiement du projet, il précise donc, dans le chapitre du même nom, le programme de travail à poursuivre et les résultats escomptés.

Ce mémoire s'articule en 6 chapitres. Dans le premier, nous introduisons le sujet par la présentation de l'entreprise et du cadre de la mission. Puis nous entrons dans le cœur du projet évolution du système VPN. Le troisième chapitre nous permet de faire l'état de l'art des technologies employées. Le quatrième chapitre aborde en détail les aspects techniques. Le chapitre suivant est consacré au déploiement du nouveau système. Enfin, le 6ème chapitre présente le bilan et les perspectives du projet.

Dans ce mémoire, le fonctionnement précis et technique de chacun des protocoles cités n'a pas fait l'objet d'un approfondissement. J'ai choisi de détailler uniquement les protocoles nécessaires à la prise de décision dans le cadre de la mise en place du nouveau VPN dans l'entreprise Netapsys.

Dans un souci de confidentialité et de sécurité, toutes les adresses IP du système d'informations de l'entreprise Netapsys évoquées dans ce document ont été remplacées.

Chapitre 1

Présentation de l'entreprise et cadre de la mission

1.1 Le groupe Netapsys

Créée en 2004, Netapsys est une *Entreprise de Service du Numérique* (ESN) spécialisée dans l'ingénierie informatique dont le siège est basé à Paris. Son activité principale est liée aux systèmes d'informations basés sur les technologies Internet (ingénierie logicielle, commerce électronique, gestion de contenu, outils collaboratifs, applications mobiles, etc.)

Implanté dans 6 grandes villes françaises (Paris, Lyon, Nantes, Lille, Rennes et Strasbourg) ainsi qu'à Madagascar, Netapsys compte aujourd'hui 430 collaborateurs. Avec un chiffre d'affaires de 26,6 millions d'euros en 2014, Netapsys réalise une croissance annuelle de 21%. Cette PME est dirigée depuis sa création par les 2 fondateurs, M. Yoann Hébert (président) et M. Jérémie Rousselle (directeur général) eux-mêmes experts en technologies objet et en systèmes d'information.

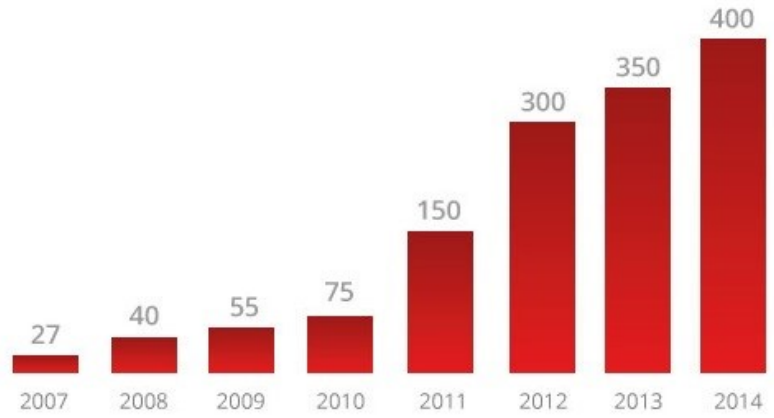


Figure 1 : Évolution du nombre de collaborateurs de Netapsys

Pour proposer des produits adaptés aux besoins des clients, Netapsys s'appuie sur des plateformes de développement riches et hétérogènes : les technologies « Open Source » (c'est-à-dire des logiciels dont la licence respecte des critères précisément établis par l'*Open Source Initiative*), de type Linux, côtoient aisément les environnements Apple Mac et Microsoft Windows. Netapsys veille ainsi à respecter les standards disponibles sur le marché afin de faire interagir ces plateformes et laisser aux collaborateurs le choix des outils à utiliser pour réaliser leurs projets.

Le site de Lyon, sur lequel je travaille, est situé en presqu'ile, à proximité de la place Ampère. L'agence fondée en 2011, actuellement dirigée par Jean-François ALEO s'appuie sur l'expertise technologique et l'esprit d'équipe de près de 70 collaborateurs répartit sur 4 pôles principaux :

- Le pôle « IBM Collaborative Solutions », spécialisé dans l'installation, la maintenance et le développement des produits IBM (7 personnes).
- Le pôle « Business Intelligence » (7 personnes),
- Le pôle « Java/.Net » qui développe les applications principalement Web pour nos clients à l'aide des technologies Java et Microsoft .Net (18 personnes)
- Et enfin un dernier pôle rassemble les collaborateurs actuellement en missions chez nos clients sur des technologies analogues à celles utilisées dans les pôles de l'agence.

En seulement 11 ans, le chiffre d'affaires du groupe Netapsys a été exponentiel grâce à une croissance externe soutenue, comme l'indique la *Figure* 2. Les compétences ainsi acquises ont permis de diversifier le savoir-faire de l'entreprise et ont contribué à sa réussite.

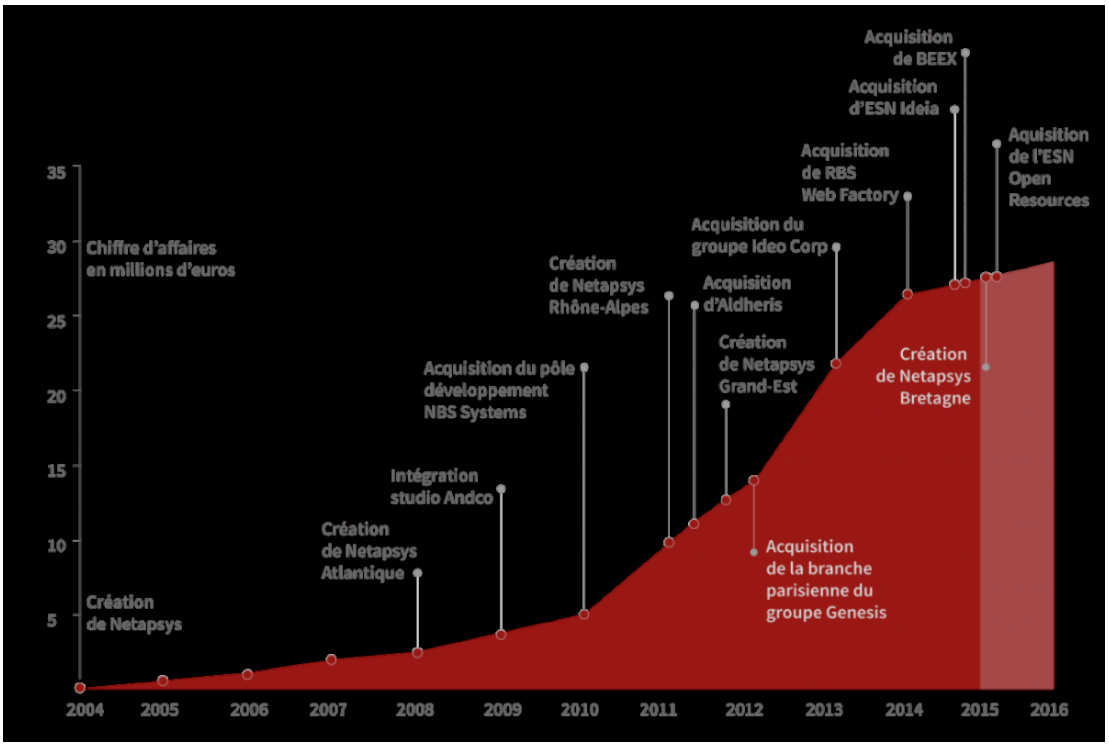


Figure 2 : Croissance du chiffre d'affaires de Netapsys et évènements clés

1.2 L'équipe « infrastructure »

Au sein de la direction des systèmes d'informations du groupe, l'équipe infrastructure est composée de 8 personnes sous la responsabilité de M. Sergio Simone. Quatre personnes travaillent sur le site parisien situé « rue de Provence » : Joyce Lambert, Dany Phy, Alexandre Veschambre, et Serdar Arikan. Trois personnes travaillent à Antananarivo (Madagascar) : Mihamina Rakotomandimby, Franck Tolojanahary et Tsirofy Rabefitseheno. En ce qui me concerne, je suis arrivé chez Netapsys le 30 septembre 2013 et je suis rattaché à l'agence de Lyon.

Nous gérons l'ensemble du parc informatique de l'entreprise, tant au niveau matériel que logiciel, y compris sur les agences où « l'équipe infra » n'est pas représentée physiquement.

Les compétences de mes collègues sont diverses et très orientées vers les logiciels Open Source. Mes connaissances sont axées sur les systèmes Microsoft Windows et je débute sur les systèmes Linux et Open Source.

L'équipe gère un peu plus de 1000 machines virtuelles (VM) pour l'ensemble de l'entreprise. La majorité de ces machines (80%) est utilisée par les développeurs pour reproduire l'environnement de nos clients et développer ainsi dans de bonnes conditions. Nous sommes chargés de l'installation et de la maintenance de ces machines virtuelles ainsi que des outils nécessaires au bon fonctionnement de l'entreprise. Pour limiter les couts de ces outils, notre infrastructure est entièrement virtualisée à l'aide des hyperviseurs *VMware ESXi*. Nous hébergeons toutes les machines de développement et de recette sur les sites de Paris - rue de Provence et Paris - rue de Mogador, Nantes, Lyon et Madagascar. Les hébergements en production, que certains de nos clients nous confient, sont hébergés dans un *Cloud Privé VMware* chez notre prestataire OVH. Ce prestataire est spécialisé dans l'hébergement informatique et propose une disponibilité quasi totale.

La mission principale de « l'équipe infrastructure » consiste à maintenir en conditions opérationnelles les outils informatiques de l'entreprise. Dans un second temps, nous réalisons le support technique auprès des utilisateurs. Parallèlement à ces missions, nous sommes chargés de réaliser des projets pour faire évoluer le système d'informations de l'entreprise. Le projet « évolution du système VPN » traité dans ce mémoire s'inscrit dans cette catégorie.

1.3 La cible recherchée : Infrastructure as Code (IAC)

Avec l'avènement de la virtualisation à tous les niveaux et des infrastructures Cloud ces dernières années, une nouvelle tendance émerge : ce procédé se nomme *Infrastructure As Code*. Cette « *bonne pratique* » consiste à rationaliser le travail des équipes chargées de l'infrastructure en automatisant au maximum le paramétrage des machines. Le principe étant de programmer le cycle de vie de ces machines afin d'agir sur les 3 axes suivants :

- **Diminution du cout** : l'automatisation permet *de facto* de créer une fois et d'appliquer à l'infini.
- **Reproductibilité** : la même programmation de l'infrastructure génère le même résultat, pas d'erreur possible contrairement à l'intervention humaine qui est faillible.
- **Changements continus** : l'évolution du système d'informations se fait par petites touches et très régulièrement. Il est possible de revenir à l'état précédent facilement.

L'infrastructure comme code se base sur des outils comme *Chef*[1], *CFEngine*[2]ou *Puppet*[3]. Chez Netapsys, nous utilisons le logiciel CFEngine pour gérer les configurations des machines sous Linux.

1.3.1.1 IAC chez Netapsys

L'équipe infra développe des modules dans le langage de CFEngine qui permettent de décrire le système d'information cible, à la manière d'un plan pour un architecte. Ces modules sont appliqués de manière cadencée (toutes les 5 minutes) sur le système cible piloté par CFEngine.

Chaque site de Netapsys qui possède un système de virtualisation, c'est-à-dire les sites de Paris (rue de Provence et rue de Mogador), Lyon, Nantes, OVH et Madagascar, est équipé d'un serveur CFEngine. Ces serveurs CFEngine se connectent sur le logiciel de gestion de versions : *Apache Subversion* (SVN en abrégé) hébergé actuellement sur le site de Provence et téléchargent la dernière version des modules développés de manière cadencée. Les machines clientes se connectent à leur

[1] https://www.chef.io/

[2] https://cfengine.com/

[3] https://puppet.com/

tour sur le serveur CFEngine le plus proche et appliquent la configuration demandée. Cette architecture possède plusieurs avantages comme l'historisation des modifications au niveau de SVN ou la tolérance aux pannes et limite aussi la quantité des données qui circule entre les sites. Voyons en détail ces avantages :

Le logiciel de gestion de versions SVN est un outil collaboratif utilisé en majorité par les équipes de développement. Il permet de rassembler le travail de plusieurs personnes en un seul point tout en assurant la traçabilité de chaque modification. De manière concrète, les utilisateurs téléchargent une copie du projet stocké sur SVN sur leur poste, apportent les modifications souhaitées et envoient en dernier lieu leur travail sur le serveur. Ces opérations peuvent être répétées à l'infini et il est simple de trouver l'auteur de chaque modification, car chacune d'entre elles est tracée dans le système.

Les 6 serveurs CFEngine récupèrent les dernières modifications sur le serveur SVN toutes les 5 minutes. Il y a donc en permanence une copie de toutes les sources SVN sur plusieurs sites ce qui permet ainsi, en cas de problème d'accès Internet, au système CFEngine de fonctionner en mode dégradé. Par ailleurs, cette copie évite aux machines virtuelles installées sur chaque site, 1000 au total, de télécharger les informations sur un serveur central, ce qui génèrerait un trafic important et une charge conséquente.

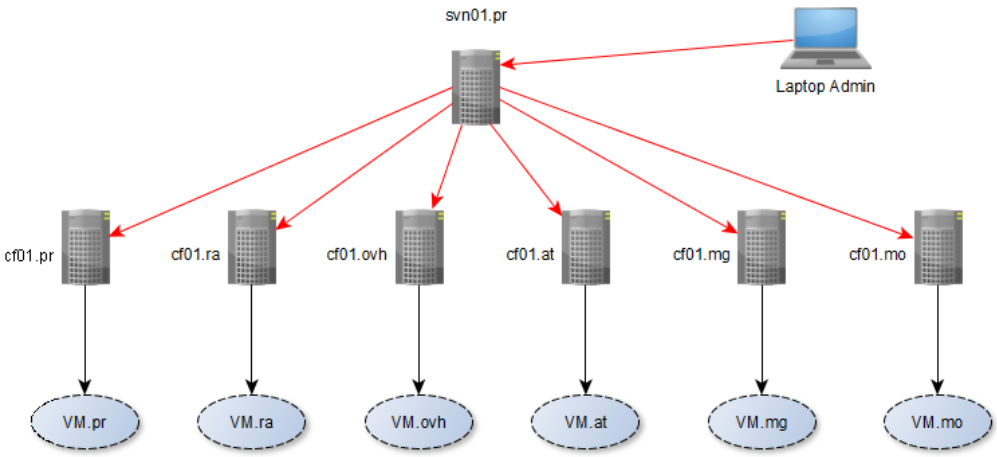


Figure 3 : Architecture de CFEngine chez Netapsys

1.3.1.2 CFEngine, un outil puissant

CFEngine est différent de beaucoup d'autres mécanismes d'automatisation puisqu'il n'est pas nécessaire de « lui dire quoi faire ». Il convient simplement de lui indiquer l'état dans lequel le système doit être, et CFEngine est capable d'appliquer automatiquement les actions à effectuer pour atteindre l'état souhaité, ou à défaut l'état le plus proche.

Les équipes en charge de l'administration des serveurs décrivent ce qu'elles souhaitent obtenir sous la forme de « promesses » théoriques, celles-ci étant des déclarations d'intention. CFEngine a pour but de réaliser ces promesses de manière à ce que l'état du système converge vers une cible « idéale » prédéfinie.

Voici un exemple de promesse : le fichier se trouvant sur le serveur central `"$(sys.workdir)/bin/file"` sera copié sur la machine cible dans le dossier `"/home/steph/tmp/test_file"`. Si ce fichier est modifié par un utilisateur ou un programme, il sera remplacé par la version présente sur le serveur central à la prochaine exécution de l'agent CFEngine :

```
bundle agent my_copy {
  files:
      "/home/steph/tmp/test_file"
      copy_from => local_cp("$(sys.workdir)/bin/file");
}
```

Pour arriver à ce but, CFEngine fonctionne en mode client-serveur. Comme nous l'avons vu précédemment, il y a 6 serveurs CFEngine chez Netapsys. Les clients sont des agents CFEngine installés sur toutes les machines Linux de l'entreprise. Les machines sous Windows ne sont pas gérées par CFEngine pour le moment, car leur nombre est encore assez faible et les configurations trop hétérogènes.

De manière cadencée, les clients CFEngine se connectent au serveur qui transmet les opérations à effectuer. Elles sont ensuite exécutées par l'agent pour atteindre l'état final souhaité. Voici une liste, non exhaustive, des opérations de base que CFEngine peut effectuer nativement :

- **Ajout d'utilisateur** : les comptes des membres de l'équipe infra sont automatiquement ajoutés sur toutes les machines Linux. Cela permet de tracer l'auteur des actions sur chaque machine.
- **Installation de logiciels** : CFEngine permet d'installer et de configurer des logiciels sur les machines Linux de façon très simple et très rapide. L'environnement de base se retrouve standardisé grâce à cette fonctionnalité.
- **Paramétrage** : les paramètres sous Linux sont majoritairement écrits dans des fichiers de configuration. CFEngine permet de forcer les valeurs souhaitées automatiquement en utilisant éventuellement un système de gabarits pour y placer des données dépendantes de chaque machine.
- **Gestion du système de fichiers** : Créer un répertoire, supprimer un répertoire/fichier, s'assurer que tel ou tel fichier/répertoire dispose de droits prévus.
- Mettre en place ou modifier **une tâche planifiée.**
- Et finalement, **lancer une commande quelconque**. On comprend bien qu'il est simplement possible de faire tout et n'importe quoi :
    - Redémarrage de la machine
    - Redémarrage d'un service
    - Initialisation d'une base
    - Import de données
    - etc.

CFEngine est un outil très puissant. Grâce à lui, le déploiement de nouvelles machines est fortement accéléré et quasiment automatisé. Notre infrastructure est homogène et les paramétrages réalisés sur les machines ne peuvent être modifiés « à la main ». Si cela se produit, l'agent CFEngine écrasera la modification à sa prochaine exécution. De plus, nous ne sauvegardons que les données utiles aux applications (bases de données, sites Web …), car, en cas de problème, nous sommes capables de recréer la machine à l'identique avec cet outil.

Pour pouvoir être compatible avec l'*infrastructure As Code* attendue chez Netapsys, l'ensemble des éléments que nous développons dans ce projet doit donc être intégré à CFEngine.

Chapitre 2

Le projet « évolution du système VPN »

2.1 Cadre du projet

2.1.1 Des enjeux importants

Les projets que nous confient nos clients font régulièrement appel à plusieurs équipes de l'entreprise qui se trouvent sur des sites distants. Notre réseau informatique est conçu pour supporter cette contrainte et propose aux développeurs d'accéder à distance aux outils que nous mettons en place. De plus, la direction technique de Netapsys souhaite répartir nos serveurs sur des sites distants pour limiter l'impact d'un problème technique qui ne pénalise ainsi qu'une partir des outils du groupe. Pour répondre à ces besoins, les interconnexions entre les sites de la société utilisent des connexions VPN.

Le projet « évolution du système VPN » est constitué de deux étapes majeures : la première consiste à faire évoluer le VPN actuel pour combler ces défauts. La seconde doit apporter une nouvelle fonctionnalité chez Netapsys : le routage dynamique. Ce routage dynamique étant totalement dépendant de l'évolution du système VPN, il est impératif de procéder dans cet ordre et de manière séquentielle pour pouvoir le résoudre.

Le *Virtual Private Network* (VPN) ou *Réseau Privé Virtuel* en français consiste à relier des lieux géographiques entre eux de manière sécurisée afin de faire communiquer leurs systèmes informatiques [Frankel, et al., 2005]. Les VPN que nous utilisons ici sont tous basés sur des connexions Internet de type fibre optique, SDSL ou ADSL. Ces interconnexions permettent par exemple de travailler sur des logiciels distants tout en conservant un niveau de sécurité optimal. Il convient de distinguer deux types d'architectures VPN : ceux qui relient des utilisateurs nomades au système informatique de l'entreprise, et les VPN site-à-site reliant les lieux géographiques entre eux. Notre besoin pour le projet relève de cette dernière catégorie, nous n'allons donc pas évoquer ici les VPN pour les utilisateurs nomades.

Chez Netapsys, le VPN est sensible, car des informations importantes y circulent : authentification des utilisateurs, habilitations, logiciels RH et comptables, outils du service infrastructure (supervision, CFEngine), GED. Cet outil doit donc être fiable et garantir une bonne sécurité des échanges. De plus, une grande disponibilité est aussi attendue : les équipes distantes doivent être à même de se connecter aux outils à tout moment de la journée.

La fonctionnalité de *routage dynamique* n'existe actuellement pas chez Netapsys. Nous souhaitons la mettre en place dans le cadre de ce projet. Cette nouvelle méthode d'aiguillage des données s'appuie intégralement sur le VPN que nous souhaitons faire évoluer.

Par opposition au routage statique qui consiste à indiquer manuellement l'adresse IP des réseaux que l'on cherche à atteindre, l'idée générale du routage dynamique est d'éviter de centraliser la configuration du routage dans les mains d'un individu. En effet, si le réseau global est complexe, la configuration peut être fastidieuse et source d'erreurs. De plus, lorsqu'un nouveau réseau est ajouté, il faut reconfigurer les routeurs. Pour prévenir tout dysfonctionnement (panne d'un routeur, ligne coupée, etc.), il faut effectuer une surveillance permanente et reconfigurer chaque routeur le cas échéant. Si la route est rétablie, il faut recommencer la manipulation.

Le facteur humain impliquant un temps de réaction plus long et des risques d'erreurs plus importants, le routage dynamique est réalisé automatiquement par les routeurs. Chaque appareil est le mieux placé pour connaitre les adresses des réseaux auxquels il est directement relié. Étant directement au contact des interfaces de communication, il peut établir un diagnostic sur l'état des liaisons. En assemblant toutes ces informations, il n'a plus qu'à les partager avec ses voisins. De proche en proche, les nouvelles se répandront à chaque routeur du réseau. L'intervention humaine ne s'opère qu'en amont dans la définition de directives et de règles à appliquer pour la diffusion des routes ; c'est ce qu'on appelle le routage dynamique.

D'autre part, le système actuel n'est pas industrialisable : toute modification nécessite une intervention humaine sur chaque routeur, par exemple lors de l'ajout d'un nouveau sous-réseau ou lors du renouvellement des certificats utilisés par OpenVPN, l'outil actuel. Celui-ci n'est pas piloté par CFEngine : nous n'avons ni historique ni traçabilité sur les opérations effectuées. Nous arrivons donc à un état du système certes fonctionnel, mais peu cohérent dans son ensemble.

Dans l'architecture actuelle du VPN, l'agence située rue de Provence a été identifiée comme un point individuel de défaillance (*Single Point Of Failure* - SPOF) parce qu'elle concentre l'ensemble du trafic VPN de l'entreprise pour le redistribuer au bon destinataire. Il s'agit d'un point critique pour l'architecture informatique, non seulement pour la gestion des incidents techniques, mais aussi dans la mesure où cette agence peut rapidement être identifiée comme une cible de choix pour mettre à mal l'ensemble de machines ou des services de l'entreprise.

Avec les derniers rachats, l'entreprise s'est considérablement étendue en termes d'agence, la mise en œuvre du routage dynamique s'avérait alors nécessaire pour les raisons évoquées précédemment (configuration manuelle source d'erreurs - industrialisation limitée). La croissance soutenue de l'effectif a dû, par ailleurs, s'accompagner d'un découpage par service pour en augmenter la sécurité (*Annexe I : Exemple de découpage d'un réseau Informatique à l'aide de VLAN*). Cette complexité accrue dans le routage actuel nous a contraints à envisager la mise en place du routage dynamique.

Enfin, la configuration du VPN ne correspond pas aux principes d'*Infrastructure As Code* attendue par l'entreprise puisqu'elle n'est pas intégrée dans CFEngine.

2.1.2 Les parties prenantes

Le projet étant interne au service Infrastructures, les parties prenantes sont toutes internes à la société Netapsys. Le *Tableau 1* suivant récapitule les différentes parties ainsi que leur rôle dans le projet.

Tableau 1 : Les parties prenantes du projet

Acteur	Fonction	Rôle dans le projet
M. Sergio Simone	Responsable infrastructures	Maitre d'ouvrage
M. Stéphane Soubeyrand	Administrateur système et réseaux	Maitre d'œuvre
M. Joyce Lambert	Administrateur système et réseaux	Conseil technique
Les autres membres de l'équipe infrastructure	Administrateur système et réseaux	Aide technique sur site pour préparation du matériel
Tous les collaborateurs de Netapsys		Clients du nouveau système VPN

Dans le cadre de mon mémoire, j'ai ainsi assuré les fonctions de maitre d'œuvre en veillant à garantir la qualité du travail et la réalisation technique. Tout au long du projet, j'ai rendu compte à M. Sergio Simone par mail et dans le cadre d'entretiens bimensuels avec la participation active de M. Joyce Lambert. Dans le cadre de ces entretiens, nous avons fait le point sur l'avancement du projet et nous avons abordé les différents problèmes rencontrés.

Afin de valoriser les compétences que j'ai acquises et de les transmettre aux 8 membres de l'équipe infra, j'ai réalisé une documentation sur le Wiki de l'entreprise relatif au fonctionnement du nouveau système et aux procédures d'utilisation.

2.2 Missions et objectifs

Le projet « évolution du système VPN » de l'entreprise Netapsys consiste à remplacer le VPN existant fonctionnant avec OpenVPN par un système plus adapté aux nouvelles contraintes.

Ce nouveau VPN s'intègre dans l'environnement existant basé sur les serveurs de type Linux – Debian. Cette nécessité s'explique par la culture de l'entreprise, très attachée aux logiciels Open Source ainsi que par les compétences de l'équipe infrastructure majoritairement orientées vers ces outils. Les serveurs concernés possèdent plusieurs fonctions : ils servent de pare-feu, de passerelle VPN et de routeur. Ces 3 rôles sont intimement liés entre eux et c'est pour cette raison qu'ils prennent place dans un équipement unique. Il y a actuellement un exemplaire de cet équipement sur chaque site de l'entreprise, soit 9 au total.

La première étape consiste à valider techniquement que le VPN est réalisable et fonctionnel avec la plateforme imposée. Pour cela, des tests probants sont réalisés en conditions aussi proches de la réalité que possible. Après cette première phase, le routage dynamique est apporté, toujours à l'aide de logiciels Open Source. Cette nouvelle fonctionnalité est intégrée à la plateforme de tests pour y être validée à son tour. Enfin, l'étape de mise en production est programmée et réalisée avec l'aide des autres membres de l'équipe infrastructure tout en minimisant l'impact de ces changements sur les outils utilisés par les collaborateurs de l'entreprise.

Le nouveau VPN tend le plus possible vers une topologie entièrement maillée, ou *Full Mesh* en anglais, dans lequel tous les sites sont reliés entre eux directement. Cette topologie améliore les temps d'accès entre les sites et évite de créer des engorgements inutiles sur les liaisons Internet des autres sites comme c'est le cas dans l'ancien VPN sur le site de Provence.

L'évolutivité du système est aussi être intégrée dans la conception du projet : l'ajout d'un nouveau site et de nouvelles routes dynamiques nécessitent le moins d'opérations possibles, ce qui augmente la fiabilité du système. La prise en compte de plusieurs connexions Internet pour un seul site est aussi une fonctionnalité intéressante à valider et à intégrer si celle-ci s'avère fonctionnelle.

Le routage dynamique, qui indique le meilleur chemin pour atteindre une destination, est complété par des fonctions de pare-feu. Le pare-feu est chargé d'autoriser le trafic légitime à l'intérieur du VPN.

Pour tendre vers la cible recherchée *« Infrastructure as Code »*, les livrables attendus sont dans le format CFEngine.

Une documentation technique est réalisée dans le Wiki. Elle est accompagnée de procédures couvrant les opérations de maintenance les plus fréquentes. Une présentation aux autres membres de l'équipe infrastructure est aussi attendue.

L'intégration avec le système de supervision actuel permet d'avoir un état de santé de système en temps réel. Des commandes renvoyant l'état des composants du système VPN sont aussi fournies. Elles sont utilisées directement sur les serveurs et donneront le statut de chaque composant.

La haute disponibilité des routeurs – pare-feu est une évolution qui intéresse Netapsys afin d'éviter les points de défaillance unique (SPOF) pour cause de panne matérielle par exemple. Le nouveau système est compatible avec cette nouvelle étape qui, dans un premier temps, n'est pas réalisée. Le trafic situé en dehors du VPN n'est pas évoqué dans ce projet.

2.3 Budget

En matière de réseau, Cisco est une référence depuis plusieurs années. Afin de confronter le cout de notre projet à des produits disponibles à l'achat, nous avons étudié la possibilité d'équiper nos sites de routeurs Cisco. Ces routeurs seraient chargés de créer les tunnels VPN, gérer le routage dynamique et assurer le rôle de pare-feu dans chacune des agences de Netapsys.

D'après mes recherches, les routeurs Cisco de la gamme 1900 (1500€ pièce environ) sont compatibles avec les sites qui ne disposent pas de serveur, à savoir Nantes, Lille et Strasbourg. Les sites de Lyon, Paris - rue de Provence, Paris - rue de Mogador, Nantes et Madagascar seraient équipés de la gamme supérieure, c'est-à-dire la gamme des Cisco 2900 (2000€ pièce). Enfin, pour relier notre *Cloud Privé* hébergé chez OVH, nous devrions opter pour une licence du logiciel Cisco CSR 1000v qui apporte des fonctionnalités analogues à un équipement physique, mais sous forme de machine virtuelle.

Nous arriverions donc à un total de (3x1500 + 5x2000 + 1x1000) = 15500€. À ce budget de base, il conviendrait d'ajouter les options requises pour améliorer la disponibilité des routeurs comme une seconde alimentation ou encore des cartes additionnelles pour équiper les agences qui possèdent plusieurs lignes Internet. Enfin, pour disposer des dernières mises à jour sur les produits Cisco, nous devrions nous acquitter d'une maintenance annuelle correspondant à environ 30% du prix d'achat initial.

Chez Netapsys, nous avions envisagé de recycler d'anciens serveurs pour effectuer les opérations de bascule vers le nouveau système VPN. Malheureusement, le nombre de ports réseau sur ces machines étant souvent limité et le cout des extensions élevé, nous avons préféré acheter des équipements adaptés à cette utilisation, mais sans système d'exploitation. Ces produits, de la marque *Lanner Electronics Inc*, s'apparentent à des petits ordinateurs de bureau, et sont intégrés dans un boitier pouvant être apposé dans nos baies informatiques. Ces équipements possèdent plusieurs ports réseau et sont évolutifs. De plus, leur cout unitaire de 1000€ sans aucune maintenance

supplémentaire est un avantage indéniable. Les machines étant livrées sans système d'exploitation, nous pouvons aisément installer une distribution Linux Debian, ce qui s'intègrera parfaitement avec l'infrastructure actuelle. Nous avons acheté 6 de ces boitiers et recyclé deux anciens serveurs. Le Cloud Privé OVH sera équipé d'une machine virtuelle sous Linux. Nous arrivons à un total de seulement 6000€.

Bien sûr, cette comparaison est tronquée, car elle ne se base que sur l'achat de matériel. Il faut aussi inclure la partie logicielle. Cisco fournit dans ces équipements son système d'exploitation maison nommé « IOS ». C'est un système d'exploitation complet et éprouvé depuis plusieurs années. Les fonctionnalités sont nombreuses, mais figées par le constructeur. La solution que nous souhaitons déployer chez Netapsys est entièrement basée sur des logiciels Open Source. Si la fiabilité de ces logiciels est aussi au rendez-vous, le cout de développement est obligatoirement supérieur face aux produits Cisco « clé en main ». A contrario, les possibilités offertes sont illimitées et ne dépendent que du savoir-faire des équipes chargées du développement.

2.4 Risques et contraintes

La première partie du projet a déjà vu le jour chez Netapsys il y a quelques années. M. Joyce Lambert a en effet déjà essayé de remplacer le VPN existant par une nouvelle version qui se rapproche fortement de ce que nous mettons en place. L'entreprise comptait à l'époque 4 sites et la partie routage dynamique n'était pas à l'ordre du jour. À la suite de plusieurs problèmes de déconnexion intempestive, le nouveau VPN a été coupé et l'ancien VPN réactivé. Par manque de temps, aucune solution n'a permis de relancer le projet jusqu'à ce jour.

MM. Sergio Simone et Joyce Lambert m'ont mis en garde sur la partie routage dynamique : il semblerait que le VPN que nous souhaitons mettre en place ne permette pas de faire circuler les données nécessaires au bon fonctionnement du routage dynamique. Une couche réseau intermédiaire gérant les trames « multicast » serait à mettre en œuvre pour contourner ce problème. Cette couche intermédiaire ajouterait un tunnel de type GRE. Ce point doit être éclairci obligatoirement pour ne pas bloquer le projet.

Enfin, la complexité générée par la nouvelle topologie *Full Mesh* ne doit pas être un frein pour effectuer les opérations de maintenance.

2.5 Planning

Le planning initial du projet du projet prévoyait une fin en décembre (*Tableau 2*). Vu que le projet n'est pas critique, celui-ci a glissé sur 2016 et la mise en production est prévue pour la fin du premier semestre.

Les principaux écarts sont explicables par les points suivants :

- Les périodes de congés des différents interlocuteurs n'ont pas été prises en compte dans le planning. Ces périodes n'ont pas permis de faire avancer le projet dans des conditions normales.
- La complexité engendrée par les tunnels GRE n'était pas intégrée au planning, ainsi que la recherche de solutions.
- Les règles de pare-feu et de routage interne ont été sous-estimées lors des premières phases du projet. Cela a engendré une surcharge de développement.

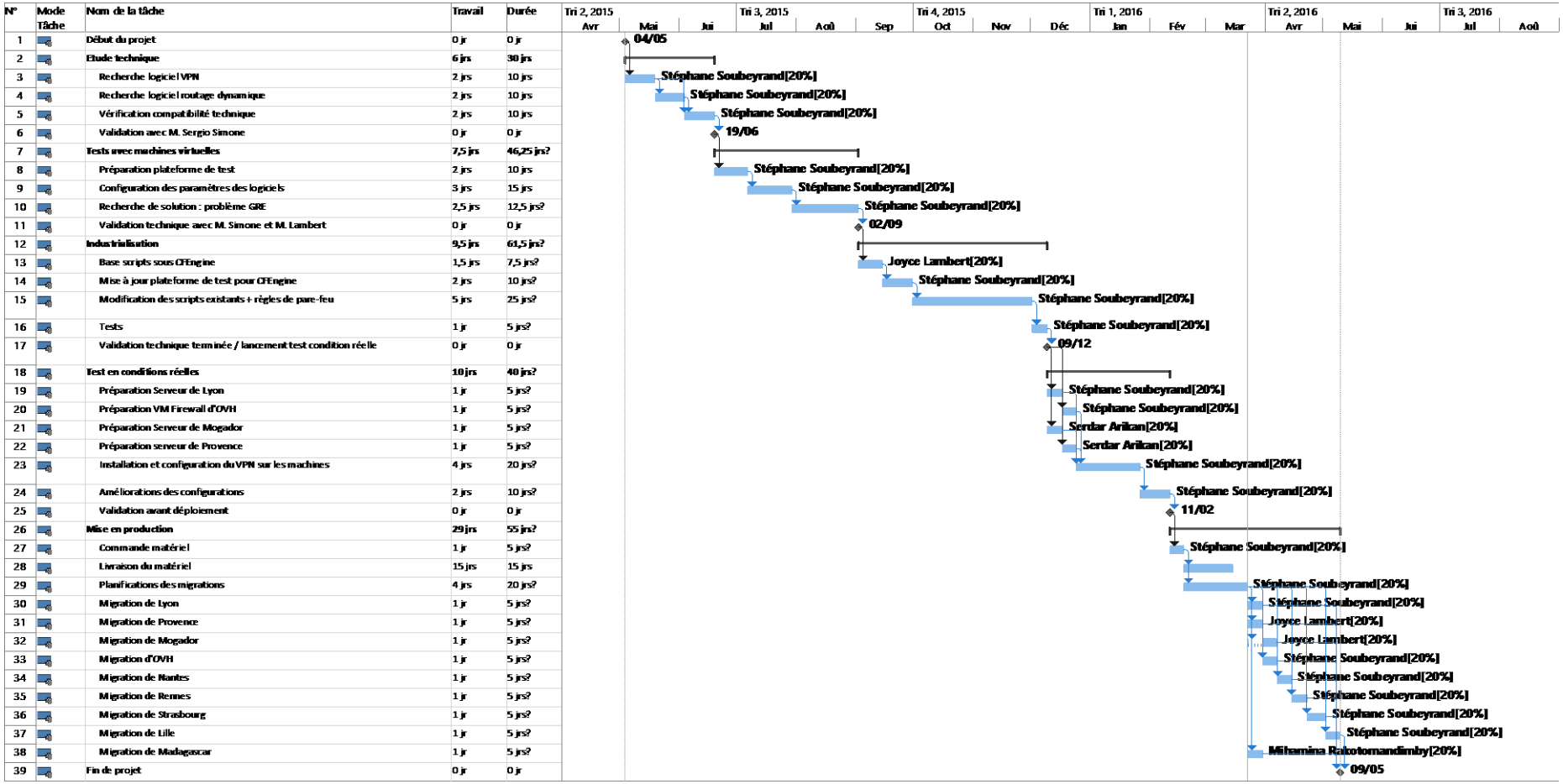
- La migration devait être réalisée vers des serveurs secondaires que nous possédons. Les prérequis de ces machines n'étant pas suffisants, nous avons dû commander du matériel supplémentaire. En ajoutant les durées de livraison, d'installation et de paramétrage de ces équipements, nous avons pris du retard de plusieurs semaines sur cette partie.

Tableau 2 : Planning du projet initial

Étape	Date	Détail
1	05/2015	Lancement du projet
2	05/2015	Laboratoire : Création d'un système VPN de test basique (3 sites)
3	06/2015	Laboratoire : Ajout d'un site + ajout de connexions Internet secondaires + test routage dynamique
4	08/2015	Laboratoire : Préparation des scripts CFEngine + Tests de configuration et de déploiement
5	09/2015	Validation par la direction
6	10/2015	Élaboration de la stratégie de migration
7	11/2015	Début de migration du VPN
8	12/2015	Fin de migration
9	12/2015	Validation globale / documentation / supervision

Le *Tableau 3* détaille le diagramme de Gantt du projet en prenant en compte les différents aléas rencontrés.

Tableau 3 : Diagramme de Gantt du projet



N°	Mode Tâche	Nom de la tâche	Travail	Durée
1		Début du projet	0 jr	0 jr
2		Etude technique	6 jrs	30 jrs
3		Recherche logiciel VPN	2 jrs	10 jrs
4		Recherche logiciel routage dynamique	2 jrs	10 jrs
5		Vérification compatibilité technique	2 jrs	10 jrs
6		Validation avec M. Sergio Simone	0 jr	0 jr
7		Tests avec machines virtuelles	7,5 jrs	46,25 jrs?
8		Préparation plateforme de test	2 jrs	10 jrs
9		Configuration des paramètres des logiciels	3 jrs	15 jrs
10		Recherche de solution : problème GRE	2,5 jrs	12,5 jrs?
11		Validation technique avec M. Simone et M. Lambert	0 jr	0 jr
12		Industrialisation	9,5 jrs	61,5 jrs?
13		Base scripts sous CFEngine	1,5 jrs	7,5 jrs?
14		Mise à jour plateforme de test pour CFEngine	2 jrs	10 jrs?
15		Modification des scripts existants + règles de pare-feu	5 jrs	25 jrs?
16		Tests	1 jr	5 jrs?
17		Validation technique terminée / lancement test condition réelle	0 jr	0 jr
18		Test en conditions réelles	10 jrs	40 jrs?
19		Préparation Serveur de Lyon	1 jr	5 jrs?
20		Préparation VM Firewall d'OVH	1 jr	5 jrs?
21		Préparation Serveur de Mogador	1 jr	5 jrs?
22		Préparation serveur de Provence	1 jr	5 jrs?
23		Installation et configuration du VPN sur les machines	4 jrs	20 jrs?
24		Améliorations des configurations	2 jrs	10 jrs?
25		Validation avant déploiement	0 jr	0 jr
26		Mise en production	29 jrs	55 jrs?
27		Commande matériel	1 jr	5 jrs?
28		Livraison du matériel	15 jrs	15 jrs
29		Planifications des migrations	4 jrs	20 jrs?
30		Migration de Lyon	1 jr	5 jrs?
31		Migration de Provence	1 jr	5 jrs?
32		Migration de Mogador	1 jr	5 jrs?
33		Migration d'OVH	1 jr	5 jrs?
34		Migration de Nantes	1 jr	5 jrs?
35		Migration de Rennes	1 jr	5 jrs?
36		Migration de Strasbourg	1 jr	5 jrs?
37		Migration de Lille	1 jr	5 jrs?
38		Migration de Madagascar	1 jr	5 jrs?
39		Fin de projet	0 jr	0 jr

Chapitre 3

État de l'art des technologies employées

3.1 Réseau Privé Virtuel (VPN)

3.1.1 Principe général

Comme nous l'avons vu au paragraphe 2.1.1, le VPN consiste à relier des lieux géographiques entre eux de manière sécurisée afin de faire communiquer leurs systèmes informatiques.

La principale alternative aux VPN consiste à louer des lignes dédiées pour relier les sites d'une entreprise entre eux. Ces connexions totalement privées sont très intéressantes, car la sécurité est déjà fournie par le loueur. Les données qui y circulent sont intégralement privées et un pirate informatique ne peut en récupérer le contenu. Comme le tarif dépend de la longueur de la ligne fournie, les entreprises implantées sur plusieurs continents par exemple, ne souhaitent pas utiliser leur budget sur ce type de connexion et préfèrent mettre à profit leur connexion Internet en créant un VPN. De plus, le nombre de liens à louer dépend du nombre de sites à relier : pour avoir un maillage *Full Mesh*, il faut que chaque site soit relié aux autres par ce type de connexion (*Figure 4 : Entreprise utilisant des connexions louées*). Il faut aussi ajouter à ce cout, prohibitif pour une majorité de sociétés, l'accès Internet « classique » nécessaire pour communiquer avec les services traditionnels.

En utilisant un VPN (*Figure 5 : Entreprise utilisant un VPN*), le résultat est équivalent et surtout beaucoup moins onéreux, à condition de bien sécuriser les échanges VPN.

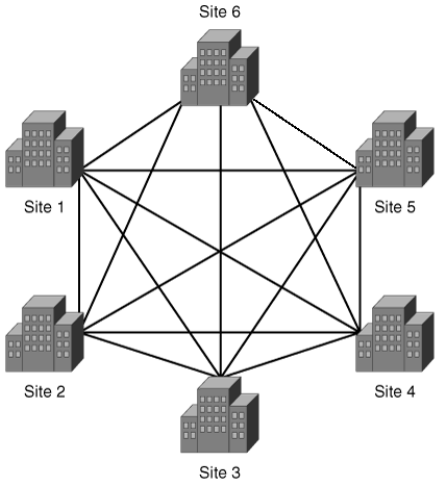


Figure 4 : Entreprise utilisant des connexions louées

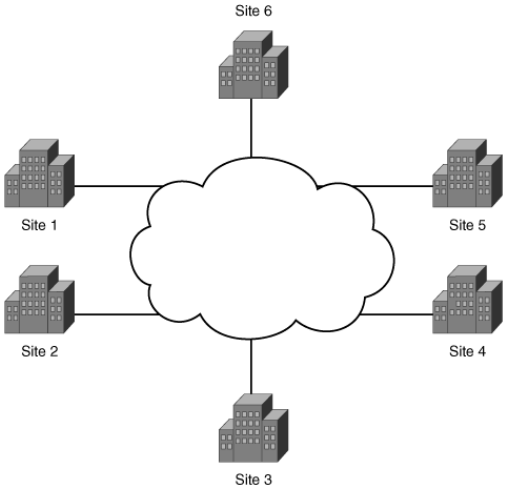


Figure 5 : Entreprise utilisant un VPN via Internet

3.1.2 Étude comparative

Plusieurs technologies sont disponibles pour créer des tunnels VPN. Nous avons choisi d'en comparer deux : OpenVPN, la technologie utilisée actuellement chez Netapsys et IPsec, qui représente la cible que nous souhaitons utiliser en remplacement. Cette étude a pour but de valider ce choix initial.

OpenVPN est un logiciel Open Source qui crée des tunnels VPN en utilisant les fonctionnalités du protocole SSL (*Secure Sockets Layer*). L'authentification est assurée par une clé partagée ou par l'utilisation de certificats. La librairie *OpenSSL* se charge de chiffrer les échanges entre les sites. Côté réseau, OpenVPN s'appuie sur UDP pour transmettre les données.

Le fonctionnement du logiciel représenté par la *Figure 6* est issu du travail réalisé par le département des technologies de l'information de l'université de Pékin [Computer Center, Peking University, 2010]. Nous pouvons voir qu'OpenVPN est une application qui fonctionne dans l'espace utilisateur, comme un logiciel classique. Celui-ci transmet les données à la carte réseau virtuelle (*TUN/TAP NIC* sur le schéma) qui est chargée de chiffrer les données avant de les transmettre aux autres applications.

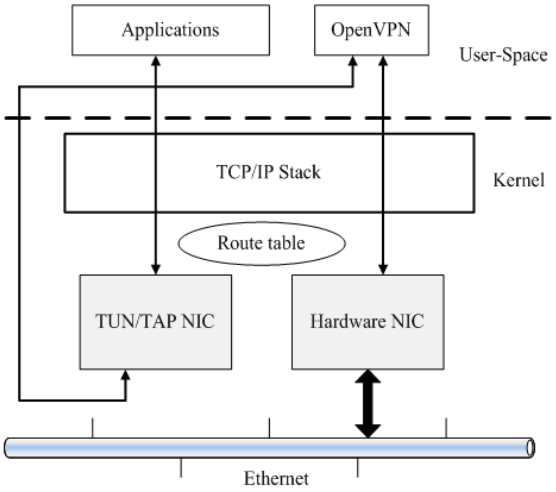


Figure 6 : Fonctionnement d'OpenVPN

Cette carte réseau virtuelle est considérée comme une carte réseau standard par le système d'exploitation : nous pouvons lui appliquer des règles de routage de la même manière qu'une carte réseau physique ainsi que des options de filtrage des paquets.

Enfin, OpenVPN fonctionne en mode client-serveur. Au lancement de la connexion, il est ainsi possible d'envoyer des règles de routage ou de spécifier l'adresse IP à utiliser. Cela se traduit par des paramétrages différents entre les clients et les serveurs à relier. Dans le cadre d'un VPN entièrement maillé, cela complexifie fortement la configuration même si avec un peu de programmation, il est possible de « générer » cette configuration où chaque pare-feu est à la fois client pour un lien et serveur pour un autre lien afin d'éviter le point de défaillance unique. L'exemple de la *Figure 7* est un aperçu de ce mode client-serveur avec 4 sites reliés en mailles pleines avec OpenVPN.

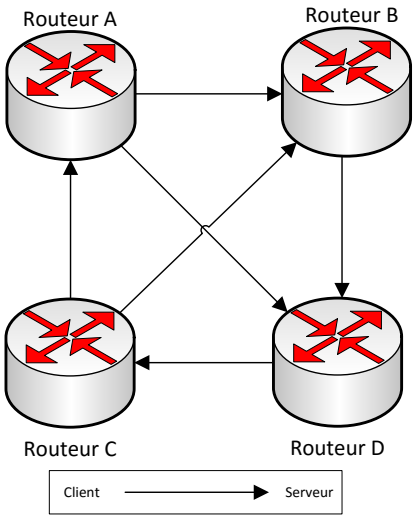


Figure 7 : Client-Serveur avec OpenVPN

Contrairement à OpenVPN qui fonctionne au niveau application (niveau 7 du modèle OSI - *Open System Interconnection*), le protocole IPsec, que nous souhaitons utiliser dans le cadre de l'évolution du système VPN de l'entreprise opère à la couche 3 de ce même modèle OSI (*Annexe II : Modèle OSI*), c'est dire au niveau réseau. Cette différence d'architecture se matérialise par une simplification de l'application, car il n'y a plus d'interface réseau virtuelle comme dans OpenVPN, mais aussi par une sécurité accrue : en fonctionnant au plus proche de la carte réseau et directement dans le noyau du système d'exploitation, les vulnérabilités et donc les failles de sécurité ou attaques informatiques *de facto* sont limitées. L'interface réseau utilisée pour créer le VPN, en général une interface reliée à Internet, est celle qui se chargera d'envoyer et de recevoir le trafic IPsec ainsi que le trafic Internet.

IPsec est une suite de protocoles validés par l'IETF (*Internet Engineering Task Force*), un groupe international qui participe à l'élaboration de standards Internet, pour obtenir un système d'échanges de données sécurisées [Frankel, et al., 2005]. Les services IPsec garantissent *l'authentification*, *l'intégrité*, le *contrôle d'accès* et la *confidentialité* des échanges. En effet, ces caractéristiques sont des exigences primordiales pour tout système VPN. Nous avons ainsi sélectionné la technologie IPsec, car elle les prend en charge nativement. Pour atteindre ces 4 objectifs, plusieurs mécanismes sont utilisés. Nous avons choisi d'en développer les principaux dans la partie *3.1.4 - Fonctionnement technique d'IPsec*. Chacun évolue au fil des années indépendamment des autres, grâce au groupe de travail de l'IETF, ce qui fait d'IPsec un protocole ouvert en constante amélioration (*Annexe III : Protocoles utilisés par IPsec*).

IPsec est supporté par une grande quantité d'équipements informatiques comme les routeurs des marques *Cisco*, *Fortinet* ou *CheckPoint*. Cette compatibilité avec les outils standard assure à Netapsys une grande évolutivité pour les années à venir.

Cette comparaison nous indique qu'IPsec peut remplacer avantageusement OpenVPN grâce à ces fonctionnalités éprouvées. Les travaux de l'IETF et l'implémentation de ce protocole dans une vaste gamme d'équipements réseau constituent une preuve de sa pérennité. OpenVPN reste malgré tout un outil très intéressant, mais de plus en plus éloigné du besoin Netapsys.

3.1.3 Fonctionnement du VPN de Netapsys

Netapsys utilisait depuis sa création un VPN basé sur OpenVPN. Le point central étant l'agence de « Paris – rue de Provence », tout le trafic intersites était obligatoirement transmis par ce nœud.

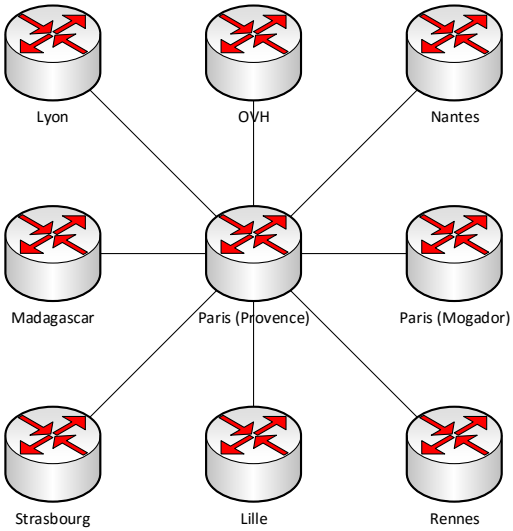


Figure 8 : Ancien VPN de Netapsys

Cette topologie engendrait une forte charge réseau sur le site de Paris – rue de Provence. De plus, en cas de problème dans cette agence, il était impossible de communiquer avec les autres sites via le VPN. Pour éviter cet écueil, Netapsys a choisi de rendre accessible sur Internet une grande partie des outils utilisés couramment par les équipes de développement, tout en limitant l'accès aux seules personnes de l'entreprise. Pour arriver à ce but, des proxys inverses (*reverse proxy* en anglais) sont installés sur les sites qui hébergent des machines virtuelles. Cependant, seules les applications de type Web sont accessibles à l'extérieur de l'entreprise. Pour communiquer avec les bases de données par exemple ou accéder à une machine virtuelle via SSH, le VPN reliant les sites est obligatoirement utilisé. Cette ouverture sur Internet limite en partie les problèmes engendrés par une coupure du VPN.

OpenVPN était le choix initial à la création de Netapsys. À l'époque, cette architecture était adaptée à l'entreprise. Cependant au fil des années, les agences se sont rajoutées dans le cadre des différents rachats ou créations de sites, mais l'architecture n'a pas évolué, rendant aujourd'hui le système VPN inadapté à la taille et aux besoins de l'entreprise.

3.1.4 Fonctionnement technique d'IPsec

Comme nous l'avons vu précédemment, IPsec est défini par l'IETF comme un cadre de standards ouverts pour assurer des communications privées et protégées sur des réseaux IP. IPsec opère au niveau de la couche réseau avec les principaux composants suivants :

- *Internet Key Exchange* (IKE) est utilisé pour négocier les paramètres de la connexion. L'authentification mutuelle des équipements souhaitant créer le tunnel, la négociation des clés secrètes ainsi que la gestion des tunnels tout au long de leur existence sont réalisées par le IKE. Il existe actuellement deux versions majeures d'IKE, la dernière, IKEv2, étant plus robuste et plus simple à implémenter.
- Deux protocoles de sécurité interchangeables : *Authentication Header* (AH) et *Encapsulation Security Payload* (ESP). AH fournit la protection de l'intégrité des données et de l'entête des trames réseaux, mais il n'apporte pas de fonctionnalité de chiffrement sur ces mêmes données. ESP chiffre les données et assure leur intégrité, mais il ne protège pas les entêtes. Néanmoins, cette protection n'est pas nécessaire dans la plupart des cas. Par conséquent, ESP est utilisé plus fréquemment qu'AH grâce à ces capacités de chiffrement. Pour les VPN qui requièrent la confidentialité des échanges, ESP est à privilégier.

Chez Netapsys, le futur VPN doit obligatoirement assurer la confidentialité des données, car elles circulent sur Internet. Le protocole « Authentication Header » est donc écarté de la suite de l'analyse.

3.1.4.1 Les protocoles AH et ESP

AH et ESP possèdent deux modes de fonctionnement, à savoir le mode *transport* et le mode *tunnel*. La différence se trouve dans le fait que le mode transport ne peut envoyer que des paquets qui sont destinés à l'hôte se trouvant au bout de la liaison IPsec. Au contraire, le mode tunnel encapsule la trame réseau dans un nouveau paquet contenant l'adresse IP de l'hôte IPsec cible vers lequel le paquet est acheminé. Avec ce mode, nous obtenons un routage statique des paquets entre deux sites. Pour que cela fonctionne, il faut déclarer à IPsec les sous-réseaux accessibles de part et d'autre du tunnel. Voici une représentation graphique des deux modes IPSEC issus de [Palomares Velasquez, 2013].

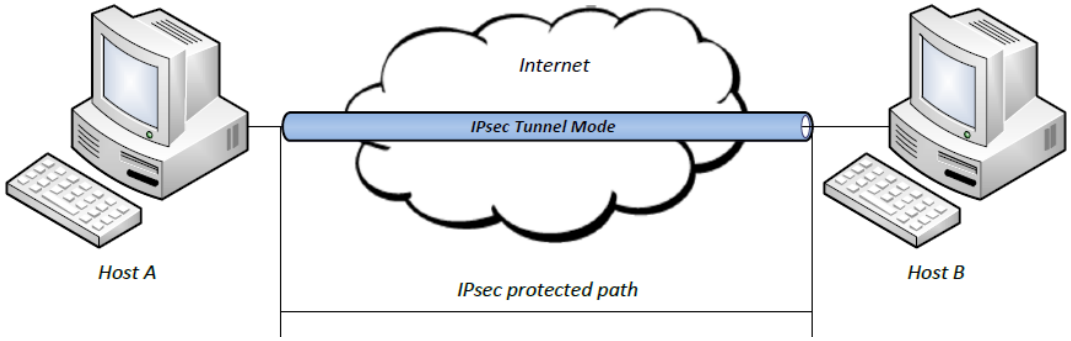


Figure 9 : Mode IPsec "transport"

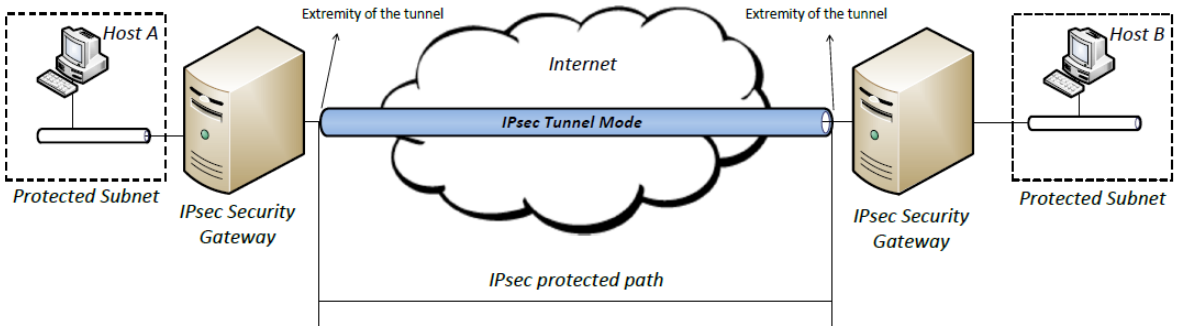


Figure 10 : Mode IPSEC "tunnel"

Les figures 11 et 12, issues du même auteur, représentent ces 2 modes au niveau du contenu des trames IP. La figure de gauche détaille le mode transport avec le protocole ESP. La figure de droite reproduit les mêmes données en mode tunnel, toujours avec ESP. La première ligne correspond au paquet original à transférer.

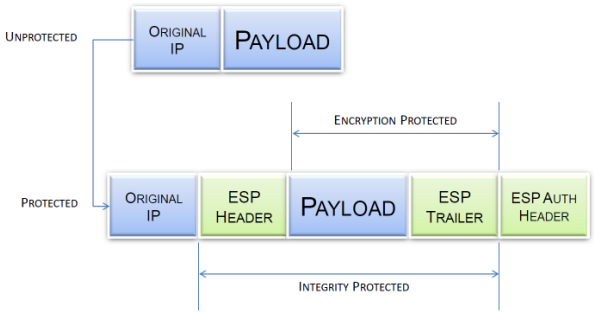


Figure 11 : Trame IP en mode IPsec "transport"

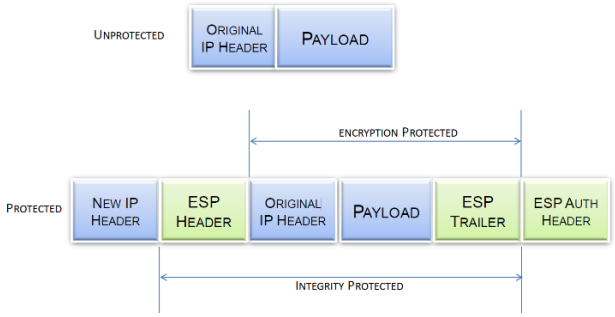


Figure 12 : Trame IP en mode IPSEC "tunnel"

Les données supplémentaires ajoutées dans le mode tunnel par rapport au mode transport sont de 20 bytes, c'est-à-dire la taille de la nouvelle adresse IP ajoutée au début de la trame IP (*Annexe IV : Trame IP encapsulé avec IPsec/ESP*). La taille d'une trame standard étant de 1500 bytes, cette surcharge représente 1,33% d'une trame. C'est peu, mais si nous souhaitons éviter ce surcoût, nous devrons utiliser le mode transport. De plus, comme nous souhaitons ajouter le routage dynamique au système VPN, le mode « tunnel » ne parait pas être la solution la plus adaptée. Le routage statique apportée par ce mode ne serait pas utilisé dans notre système.

Dans cette analyse technique du protocole IPsec, il apparait très clairement que nous avons besoin d'utiliser le mode transport d'IPsec pour maximiser les performances du VPN. Couplée au protocole de chiffrement fourni par ESP, la couche sécurité attendue pour ce projet est satisfaite.

3.1.4.2 Fonctionnement d'IKEv2

Les services de sécurité d'IPsec utilisent un chiffrement basé sur des clés symétriques. Il est nécessaire pour les deux équipements se trouvant de chaque côté du tunnel à créer de se mettre d'accord sur le mécanisme à utiliser pour partager la clé secrète ainsi que sur les clés pour gérer l'authentification et le chiffrement des données. D'après le standard édité par l'IETF sous le numéro RFC4306[4], IKEv2 combine les concepts d'authentification, de gestion des clés, et d'association de sécurité (SA) pour établir la sécurité requise pour assurer les communications privées. Cette norme définit les procédures et les formats de paquets à utiliser pour établir, négocier, modifier et supprimer des associations de sécurité. IKE fournit ainsi un cadre d'échange cohérent tout en restant indépendant de la technique de génération de clé, de l'algorithme de chiffrement et des mécanismes d'authentification. Cette indépendance vis-à-vis des algorithmes employés est l'atout majeur d'IKE. Il correspond ainsi au socle de gestion des tunnels IPsec et autorise un remplacement des algorithmes afin de suivre les dernières évolutions en matière de cryptographie.

IKE comporte deux phases, la première qui génère l'*IKE Security Association* (IKE SA) et la seconde, qui a pour but de créer des *IPsec Security Association* (IPsec SA). L'IKE SA est le résultat des premiers échanges entre les hôtes qui souhaitent établir le tunnel IPsec. Ces échanges sont représentés dans la figure suivante issue de [Palomares Velasquez, 2013]. Les échanges dans la phase 1 utilisent l'algorithme de *Diffie-Hellman* qui représente un standard pour mettre en place un secret partagé. Dans notre cas, ce secret est une clé symétrique qui est utilisée pour chiffrer et déchiffrer les messages. La phase 2 se charge de décrire le tunnel à mettre en place en spécifiant par exemple l'algorithme de chiffrement à utiliser (AH ou ESP).

[4] http://tools.ietf.org/html/rfc4306

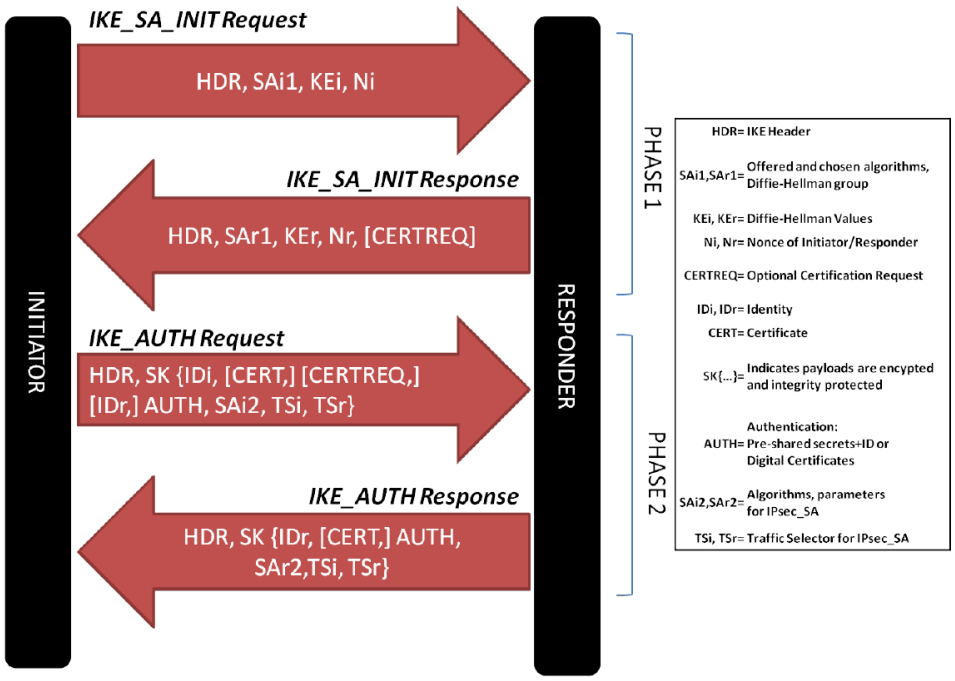


Figure 13 : IKE : les échanges de données lors de l'établissement de la connexion

D'autres informations supplémentaires sont échangées durant ces phases d'établissement du tunnel comme l'algorithme de chiffrement à utiliser, les délais de renouvellement des clés ou la méthode pour vérifier que le tunnel est bien disponible.

Vu qu'ils ne sont pas critiques et n'impactent pas l'architecture globale du système VPN, ces paramètres sont configurés lors des premiers tests.

En conclusion, nous avons vu dans cette section, l'orientation générale de la configuration que nous souhaitons mettre en place. Ces recherches mettent en lumière la complexité des protocoles nécessaires à la création d'un espace d'échange de données sécurisées fonctionnant avec IPsec. Une erreur de conception n'est surement pas bloquante, mais peut diminuer fortement les performances ou compromettre la sécurité des échanges.

3.2 Le routage dynamique

3.2.1 Intérêt de cette technologie

Nous avons évoqué les principaux défauts du routage statique lors de la définition des enjeux du projet (*2.1.1 - Des enjeux importants*). Le facteur humain étant la principale source de problème pour obtenir un environnement fiable et fonctionnel. A contrario, le routage dynamique est réalisé directement par les équipements réseau, l'intervention humaine se limite à la déclaration initiale des routes sur l'équipement le plus proche. La propagation des données est automatisée et l'erreur humaine éliminée.

Nous avons choisi d'étudier les protocoles de routage dynamiques les plus courants : RIP (Routing Information Protocol), BGP (Border Gateway Protocol), EIGRP (Enhanced Interior Gateway Routing

Protocol) et OSPF (Open Shortest Path First). Cette comparaison a pour but de sélectionner le protocole le plus adapté aux besoins et à l'environnement de Netapsys.

On classe ces protocoles en deux catégories selon leur champ d'application. Les protocoles IGP (*Interior Gateway Protocol*) dont font partie RIP, EIGRP et OSPF et les protocoles EGP (*External Gateway Protocol*) dont le représentant le plus courant est BGP. Pour comprendre ce classement, il convient de définir la notion de Système Autonome (*Autonomous System - AS* en abrégé) : chaque système autonome est un ensemble de réseaux informatiques sous le contrôle d'une entité unique et qui partage une politique de routage interne cohérente, d'après [Sequeira, 2013]. L'entité unique représente par exemple un fournisseur d'accès à Internet ou une entreprise de très grande taille. Les protocoles IGP sont utilisés à l'intérieur de ces AS, les protocoles EGP sont destinés à relier les AS entre elles, comme indiqué sur la *Figure 14*.

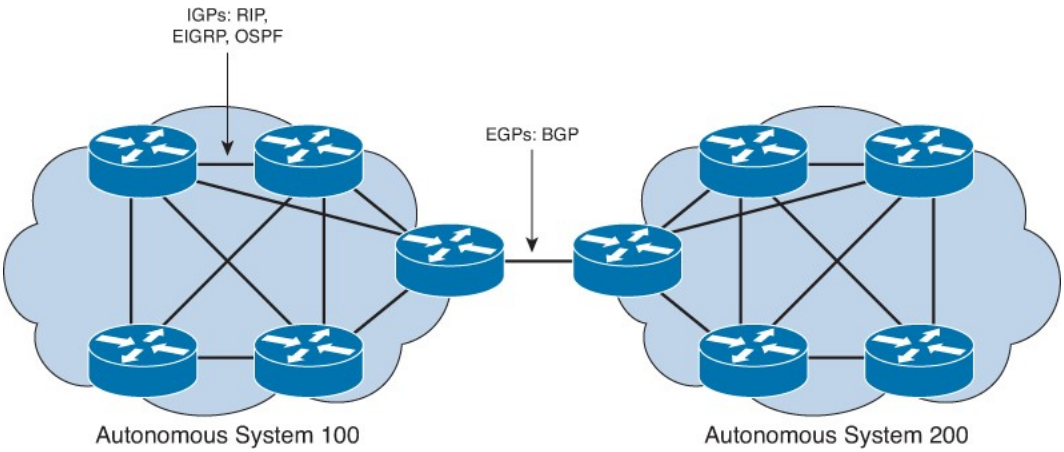


Figure 14 : Types de protocoles de routage dynamique [Sequeira, 2013]

De plus, les protocoles de routage dynamique sont aussi classés par la manière dont ils opèrent :

- Vecteur de chemin (RIP, BGP) : l'approche du routage par vecteur de distance détermine la direction et la distance pour chaque lien dans le réseau. La distance est définie comme une métrique et la direction donne le nom du routeur suivant. Chaque routeur apprend donc les routes par ses routeurs voisins.
- État de lien (OSPF) : L'approche par état de lien, aussi appelée *Shortest Path First* (*SPF - Chemin le plus court*), crée une image en mémoire de la topologie exacte du réseau. Cette image est partagée avec ces voisins pour obtenir une image globale du réseau.
- Hybride (EIRGP) : l'approche hybride combine les algorithmes de l'état de lien et des vecteurs de chemins.

En définitive, il n'existe pas de meilleur algorithme de routage dynamique, chacun étant adapté à une situation particulière. Nous avons choisi d'éliminer le protocole BGP du reste de l'analyse, car il ne correspond pas au besoin initial : ce protocole est bien plus adapté pour fonctionner à l'extérieur d'un système autonome alors que le but du projet est d'en créer un.

Le *Tableau 4* ci-dessous compare les principales caractéristiques des protocoles IGP. Il est réalisé à partir des données du site Internet *inetdoc.net* [Latu, 2015].

Tableau 4 : Comparaison des protocoles de routage dynamique

Caractéristique	RIP	OSPF	EIRGP
Type de protocole	Vecteur	SPF	Vecteur + SPF
Métrique utilisée	Nombre de sauts	Cout du lien	Bande passante / Délai
Version en cours	Version 2	Version 2 pour IPv4 Version 3 pour IPv6	« Draft » (brouillon)
Date de dernière modification	1998	2015	2016
Transport	UDP 520	IP Protocol 89	IP Protocol 88
Durée de convergence	Lente	Rapide	Très rapide
Minuteries de mise à jour	30 sec	Sur changement	Sur changement
Adresse de mise à jour	Multicast 224.0.0.9	Multicast 224.0.0.5 224.0.0.6	Multicast 224.0.0.10

D'après ce tableau, le protocole EIGRP possède beaucoup de qualités. Cependant, Cisco n'a ouvert son fonctionnement à l'IETF en vue d'en faire une norme il n'y a que peu de temps (2013) et n'est actuellement qu'a l'état de brouillon [Datatracker IETF, 2016]. Il est donc difficile d'en trouver une implémentation en dehors du matériel Cisco.

L'utilisation des logiciels Open Source étant un prérequis au projet, nous avons choisi d'utiliser le protocole OSPF pour notre VPN pour sa capacité à créer un réseau dynamique plus rapidement que le protocole RIP.

3.2.2 Le fonctionnement d'OSPF

D'après [Shamim, et al., 2002], le développement a commencé en 1987, et OSPF version 2 a été créé en 1991 avec la RFC1247[5]. L'objectif était d'avoir un protocole d'état de liaison qui est plus efficace et évolutif que RIP. La RFC5340 [6](juillet 2008) est la dernière version majeure d'OSPF, elle est dédiée au support d'IPv6.

OSPF fonctionne au-dessus d'IP et utilise le numéro de protocole 89, tout comme TCP fonctionne au-dessus d'IP et utilise le numéro de protocole 6. OSPF n'utilise pas de protocole de transport, tel que TCP, pour la fiabilité. Le protocole lui-même dispose d'un mécanisme de transport fiable.

OSPF est un protocole de routage sans classe qui prend en charge les masques de sous-réseau de taille variable (*variable-length subnet masking* – VLSM). OSPF utilise des adresses multicast (224.0.0.5 et 224.0.0.6) pour envoyer les informations nécessaires à son fonctionnement.

[5] https://tools.ietf.org/html/rfc1247

[6] https://tools.ietf.org/html/rfc5340

Pour mémoire, les trames multicast envoyées par un émetteur seront reçues par un ou plusieurs destinataires qui sont à l'écoute de ces trames. Le broadcast est reçu par tous les équipements dans le même sous-réseau, sans distinction. Enfin, unicast est le plus employé sur nos réseaux actuels : il s'agit de la communication entre deux hôtes identifiés. Les adresses unicast sont utilisées par exemple par les protocoles FTP, SSH, ou HTTP comme l'illustre la *Figure 15*.

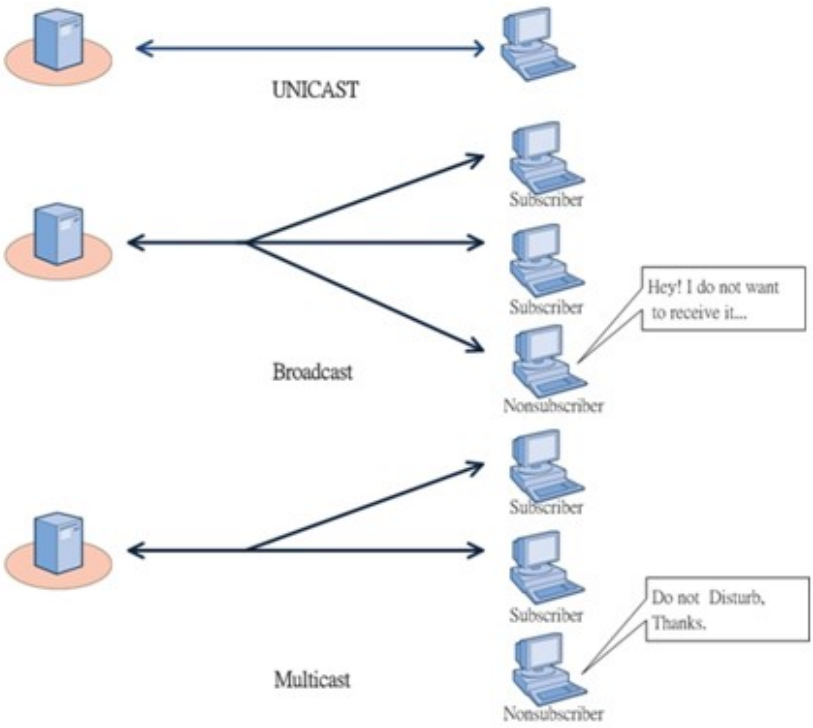


Figure 15 : Unicast, Broadcast et Multicast

3.2.2.1 *L'algorithme de Dijkstra*

Comme nous l'avons vu au paragraphe précédent, OSPF est un protocole à états de lien. Il utilise l'algorithme de *Dijkstra*. Cet algorithme permet de résoudre le problème *du plus court chemin*. Il permet, par exemple, de déterminer le chemin le plus court pour se rendre d'une ville à une autre en connaissant le réseau routier de la zone concernée. Avec l'algorithme de Dijkstra, le poids du chemin entre deux liens est la somme des poids des arcs qui le composent.

L'exemple suivant, issu de [Shamim, et al., 2002], illustre le fonctionnement de l'algorithme :

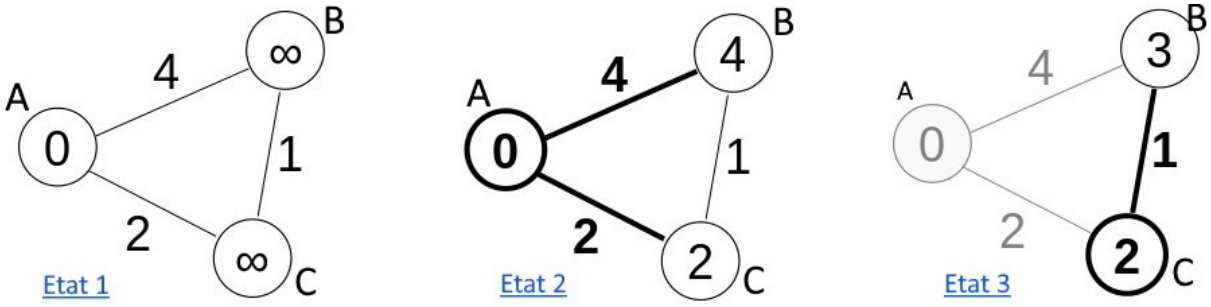


Figure 16 : Algorithme de Dijkstra

État 1 : Nous sommes placés au point A et souhaitons connaitre le cout pour aller en B et en C en sachant que le lien A-B coute 4, le lien A-C coute 2 et que le lien B-C coute 1. Dans un premier temps, l'algorithme initialise les cibles B et C avec une mesure infinie.

État 2 : En partant de A pour aller en B, cela coute 4. Comme 4 est plus petit que le signe infini déjà inscrit, nous inscrivons 4 dans la case B. Toujours en partant de A et pour aller en C, cela coute 2. Comme 2 est plus petit nous inscrivons 2 dans la case C.

État 3 : La dernière étape consiste à additionner le chemin B-C aux chiffres déjà collectés pour vérifier si le chemin n'est pas plus court en utilisant cette section : pour aller à B, nous avons déjà un cout direct de 4. Si nous continuons notre chemin vers C, le cout de A-C est calculé de la manière suivante : A-B + B-C= 4 + 1 ce qui nous donne un cout total de 5. Vu que le cout est supérieur à celui déjà inscrit dans C (5>2), aucune modification n'est effectuée. En revanche, pour aller à B en passant par C, nous obtenons un cout de 3 (A-C + C-B = 3). Il faut donc remplacer le cout inscrit en C par 3.

Au final, grâce à l'algorithme de Dijkstra, nous obtenons une carte complète du réseau avec les meilleurs chemins à utiliser pour se rendre sur chacun des sommets possibles. Dans notre exemple, le lien entre A et B n'est pas utilisé, car son cout est supérieur à celui du chemin A-C-B. Cependant, si le point C devenait indisponible ou que le cout des liens évolue, il faudrait relancer l'algorithme de Dijkstra pour recréer la représentation incluant ces modifications.

3.2.2.2 Les échanges d'OSPF

D'un point de vue général, le mode opératoire d'OSPF est le suivant :

1. Tous les routeurs OSPF envoient des paquets « *HELLO* » sur les interfaces activées au préalable. Si deux routeurs OSPF partagent une connexion permettant de transmettre ces informations, ils recevront leurs paquets *HELLO* respectifs. À ce moment-là, ils s'enregistrent mutuellement comme voisins (*neighbors*) et élisent un maitre sur le réseau s'il n'existe pas.
2. Ces routeurs contigus créent ainsi des liens virtuels points à point. OSPF définit plusieurs types de réseaux et plusieurs catégories internes de routeurs. La mise en place d'une contiguïté est déterminée par les types de routeurs échangeant les paquets Hello ainsi que le type de réseau sur lequel transitent les données.
3. Chaque routeur envoi à ces voisins des trames LSAs pour *link-state advertisements*. Les paquets LSAs décrivent toutes les connexions que possède le routeur ainsi que la liste des routeurs voisins. L'état et le cout de chaque lien sont aussi transmis.
4. Chaque routeur enregistre ces informations dans sa base d'états de liens et transmet une copie de ses informations à tous ces autres voisins.
5. Après ces transmissions cascadées, tous les routeurs de la zone possèdent un exemplaire complet de tous les réseaux qui composent la zone.
6. Enfin, les routeurs utilisent l'algorithme de Dijkstra et calculent un plan du réseau le plus optimisé possible, en se mettant à l'origine de l'arbre OSPF ainsi généré.
7. De manière cadencée, les routeurs s'envoient des paquets HELLO pour vérifier que les liens sont toujours actifs. Si une connexion ne répond pas, le processus recommence à la troisième étape en éliminant cette connexion des paquets LSAs transmis. Un nouvel arbre est créé en ne prenant pas en compte le routeur injoignable.

3.2.3 Les échanges réseau « OSPF »

Les protocoles à vecteur de distance comme RIP utilisent aveuglément le broadcast ou le multicast en envoyant sur chaque interface toute leur table de routage toutes les 30 secondes (par défaut). A contrario, les routeurs OSPF comptent 5 différents types de paquets pour identifier les voisins et mettre à jour les informations de routage à état de lien [Shamim, et al., 2002].

Tableau 5 : Types de paquets OSPF

N°	Type	Description
Type 1	Hello	Établis et maintiens les informations de contiguïté (adjacency information) avec les voisins.
Type 2	Database Description packet (DBD)	Décris le contenu des bases de données d'état de liens (link-state database) des routeurs OSPF.
Type 3	Link-state request (LSR)	Demande des éléments spécifiques des bases de données d'état de liens (link-state database) des routeurs OSPF.
Type 4	Link-state update (LSU)	Transporte les link-sate advertisements, les LSA, aux routeurs voisins.
Type 5	Link-state acknowledgment (LSAck)	Accusés de réception des LSA des voisins

3.2.3.1 Les paquets « Hello »

Les paquets *Hello* sont les premières informations OSPF qui transitent sur le réseau (*Annexe V : Structure complète du message OSPF/Hello*). Ces paquets ont pour but d'élire un serveur maitre (*designated router - DR*) et un serveur maitre de secours (*backup designated router - BDR*).

Tous les autres routeurs sur le même segment envoient leurs informations d'état de lien au DR. Le DR agit comme porte-parole pour le segment. Le DR se chargera ainsi de renvoyer les informations d'état de lien à tous les autres routeurs du segment avec l'adresse multicast 224.0.0.5. Cette technique a pour but de diminuer les échanges dans les grands réseaux OSPF.

Malgré tout le bénéfice en efficacité de cette procédure d'élection, il y a un inconvénient : le DR sera un point de défaillance unique. Pour éviter cela, un second routeur est aussi élu, il deviendra maitre en cas de problème sur le routeur principal. Pour être sûr que les DR et BDR voient l'état de lien de tous les routeurs sur le segment, l'adresse multicast 224.0.0.6 est utilisée pour *tous* les routeurs désignés (DR). Les autres routeurs qui ne sont ni DR ni BDR sont appelés *DROTHER*.

Dans le VPN de Netapsys, nous souhaitons éviter ces problèmes de point de défaillance unique. OSPF possède un paramètre qui associe un poids à chaque routeur. C'est le routeur qui possède le poids le plus important qui est élu DR. En fixant le poids de chaque routeur pour l'élection à zéro, nous avons annulé cette étape d'élection. Pour mémoire, nous souhaitons mettre en place un réseau entièrement maillé, ce qui se traduit par une connexion directe de chaque routeur à tous les autres via un lien VPN. Dans notre situation, élire un routeur maitre n'apporte aucun gain et peut, bien au contraire, amener un dysfonctionnement temporaire en cas de panne de ce dernier.

3.2.3.2 Les zones OSPF

Pour gérer les réseaux de plus en plus grands, OSPF intègre une fonction de découpe en zones (*AREA* dans le langage OSPF) dont nous pouvons voir une représentation avec la *Figure 17*. La zone 0 est employée comme zone principale, c'est la colonne vertébrale du réseau. Les autres zones sont indépendantes et assurent un routage dynamique de la même manière que dans la zone 0. Les routeurs à cheval sur plusieurs zones sont appelés *Area Border Router.* Ils possèdent les données de tous les réseaux auquel ils sont connectés, contrairement aux routeurs qui se trouvent à l'intérieur d'une zone, qui ne possèdent que les données relatives à la zone à laquelle ils appartiennent.

Ce découpage a pour but d'accélérer la convergence du réseau en limitant le nombre de routeurs à synchroniser dans une zone. En général, les zones OSPF correspondent à des régions géographiques, mais il est possible de créer des topologies plus complexes.

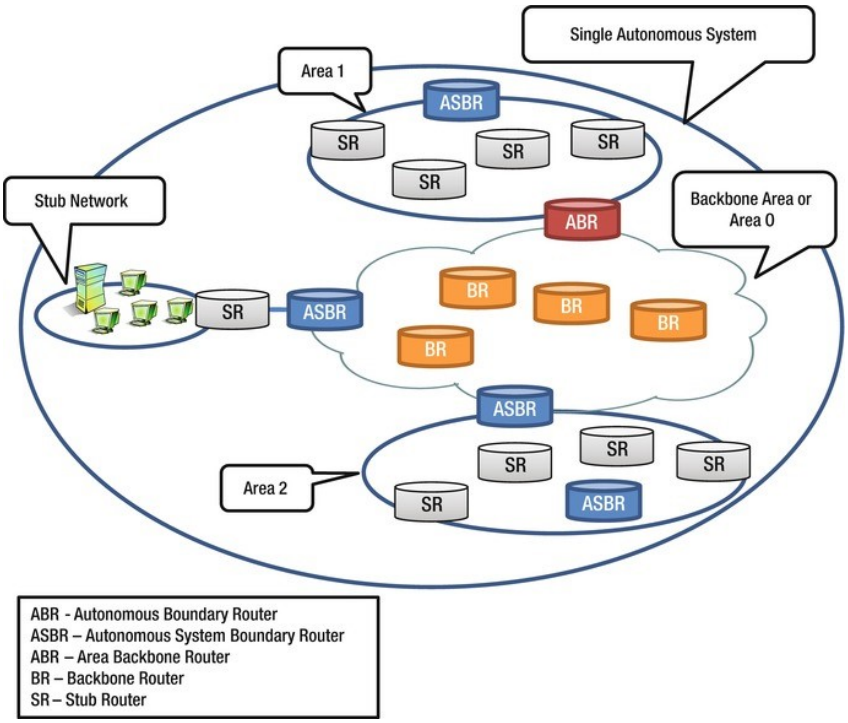


Figure 17: Zones OSPF [Tadimety, 2015]

Dans le cas de Netapsys, nous avons envisagé de créer une zone principale contenant les sites qui possèdent un système de virtualisation et une zone pour les autres sites. Tous les routeurs des sites de la zone 0 sont déclarés en tant que ABR. Cette configuration n'a pas été retenue, car elle n'apporte aucune fonctionnalité additionnelle et complexifie le paramétrage du système.

Nous avons donc créé une seule zone OSPF. Cette zone contiendra tous les routeurs de l'entreprise.

3.3 Les tunnels GRE

La conception d'un VPN en utilisant IPsec a cependant des limites :

- IPsec peut chiffrer / déchiffrer uniquement le trafic IP.
- Le trafic IP destiné à une adresse IP de multidiffusion (multicast) ou de diffusion (broadcast) ne peuvent pas être traitées par IPsec, ce qui signifie que le trafic multicast ne peut pas traverser le tunnel IPsec.

En outre, de nombreux protocoles de routage (comme EIGRP, OSPF et RIPv2) utilisent une adresse de multicast ou de broadcast pour communiquer. Par conséquent, le routage dynamique utilisant ces protocoles de routage ne peut pas être configuré entre les sites reliés par IPsec. Ces difficultés peuvent être surmontées par la configuration d'un tunnel GRE (*Generic Routing Encapsulation*) qui sera à même de transmettre les informations non supportées par IPsec entre les sites.

GRE est un protocole ouvert, initialement développé par Cisco, qui peut donc être mis en place sur des plateformes différentes. Il est défini par la RFC2784[7]. Il est possible depuis un hôte donné d'ouvrir plusieurs tunnels GRE, qui sont conçus pour encapsuler n'importe quel protocole de niveau 3 dans IP. Pratiquement, le plus souvent, de l'IP dans de l'IP. Malheureusement, ce n'est pas un protocole sécurisé d'une part, parce que GRE ne prévoit pas de chiffrement des données qui passent dans le tunnel (il n'est pas étanche), d'autre part parce que GRE ne prévoit pas l'authentification des extrémités du tunnel.

Ces faiblesses ne sont cependant pas critiques, car en associant les tunnels GRE à l'intérieur des tunnels IPsec, nous obtenons un système d'échanges de données sécurisé supportant les communications multicast et donc le routage dynamique.

Nous avons donc décidé de mettre en place un tunnel (GRE) dans un autre tunnel (IPsec). Avec GRE, les échanges n'étant pas sécurisés, la configuration des tunnels est très rapide : il faut seulement indiquer les adresses IP publiques sur lequel GRE doit être créé et ajouter des adresses IP privées qui correspondront aux points d'entrée et de sortie du tunnel GRE.

L'encapsulation des données dans un tunnel GRE ajoute 24 bytes supplémentaires aux données originales. Ces ajouts sont situés uniquement au niveau de l'entête et de l'adresse IP supplémentaire destinée à l'autre extrémité du tunnel (*Annexe VI: Entête d'un paquet GRE*).

3.4 Technologies retenues

Au final, nous obtenons 3 couches pour créer notre VPN à routage dynamique :

- IPsec, qui est chargé de transporter les données entre les sites de manière sécurisés
- GRE transporte les données du routage dynamique ainsi que les données des utilisateurs.
- OSPF s'occupe de créer une couche de routage dynamique de tout le réseau de l'entreprise et utilise le tunnel GRE pour échanger avec les autres routeurs.

La *Figure 18* indique le cheminement du paquet original pour arriver au but, c'est-à-dire de l'autre côté du tunnel VPN. Tout d'abord, nous pouvons voir l'encapsulation des données dans le tunnel GRE, puis dans un deuxième temps, toutes les informations sont chiffrées par IPsec pour être transférées à la destination via Internet. À l'arrivée, les étapes sont exécutées dans l'ordre inverse : déchiffrement IPsec, suppression des informations GRE et enfin, le paquet est transmis à la cible souhaitée.

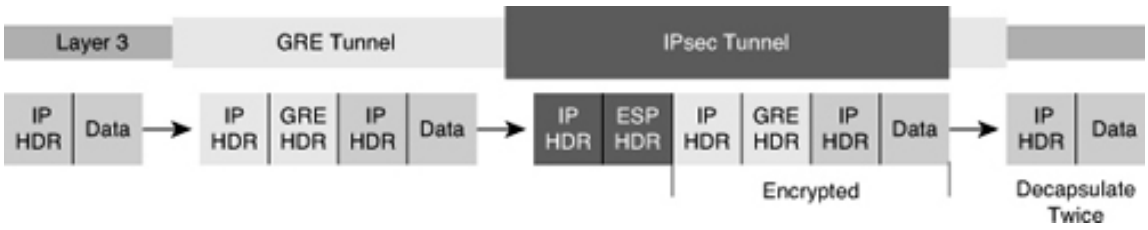


Figure 18 : Encapsulation des données via GRE puis via IPsec

[7] https://tools.ietf.org/html/rfc2784

Les trames réseau qui circulent sur Internet possèdent l'architecture représentée par la *Figure 19*. Nous pouvons la décrire de la manière suivante :

- Les données à transférer se trouvent dans la partie « *Orig. Payload* » (charge utile originale). Ces données sont accompagnées de l'adresse IP de destination d'origine « *Orig IP Hdr* »
- Les informations précédentes sont encapsulées dans le tunnel GRE (« *GRE Hdr* »)
- Les données GRE sont chiffrées par IPsec, puis envoyés sur le réseau en ajoutant l'adresse IP de destination (« *IP Delivery Hdr* ») et les infos complémentaires d'IPsec (« *ESP Hdr* », « *ESP Trailer* » et « *ESP Auth* »)

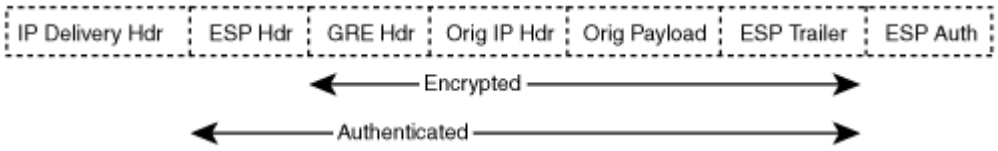


Figure 19 : Paquet IPsec + GRE

Une remarque s'impose sur cette architecture : pourquoi faire transiter les données des utilisateurs via les tunnels GRE et pas directement via IPsec ? Cette limitation est liée au routage dynamique : les routeurs OSPF utilisent les tunnels GRE pour échanger leurs données. Ils notent dans leur table de routage le lien utilisé pour atteindre chaque autre routeur OSPF. Ce lien étant le tunnel GRE, les routes qu'ils vont ainsi mettre en place utiliseront aussi les tunnels GRE.

Au final, notre étude nous apporte les protocoles nécessaires à la réalisation de notre projet. Les protocoles sélectionnés, IPsec, OSPF et GRE correspondent en tout point au besoin de l'entreprise Netapsys : ils ont fiables, performants et matures. Leur licence Open Source assure une pérennité ainsi qu'une compatibilité avec les outils que nous utilisons. Dans le chapitre suivant, nous décrivons comment ces protocoles sont paramétrés sur nos plateformes de tests.

Chapitre 4

Réalisation technique

4.1 Maitrise de la technologie

4.1.1 Logiciels IPsec

Comme nous l'avons vu au chapitre précédent, le protocole le plus adapté pour réaliser les tunnels VPN est IPsec. Par ailleurs, le cahier des charges du projet nous impose d'utiliser la plateforme Linux. Après avoir effectué des recherches, nous avons sélectionné 4 logiciels compatibles avec ces critères.

Le plus ancien, qui se nomme *FreeS/WAN*, n'a pas été mis à jour depuis 2004. Il n'est actuellement plus maintenu, nous ne l'avons donc pas retenu. *Openswan* et *StrongSwan* sont deux logiciels créés à partir du code source du logiciel *FreeS/WAN* ; nous nommons cette pratique en informatique un *fork*.

Openswan[8] est passé sous le contrôle de la société *Xerelance* en 2012. Cette société propose un support technique payant tandis que la communauté s'est considérablement amoindrie ces dernières années. Il est donc difficile de trouver une communication récente sur ce logiciel. Suite à cette prise de contrôle, une partie des développeurs a *forké* le logiciel en raison de divergences d'opinions sur l'évolution de ce dernier. C'est ainsi que *Libreswan* a vu le jour.

Libreswan[9] correspond à notre besoin pour créer les tunnels IPsec. Cependant, ce logiciel n'est pas inclus dans les dépôts officiels de Debian. Il n'a donc pas été testé en profondeur par les équipes qui développent le système d'exploitation Debian.

StrongSwan[10] est actuellement maintenu par Andreas Steffen, professeur de sécurité dans les communications de l'université des sciences appliquées de Rapperswil en Suisse. Ce logiciel est présent dans les dépôts Debian et une communauté active est présente sur les forums officiels.

Ces recherches nous ont permis de sélectionner le meilleur candidat pour note système VPN IPsec : c'est le logiciel StrongSwan qui est le plus à même de monter les tunnels entre les sites de l'entreprise Netapsys grâce à son respect des standards IPsec et ces mises à jour régulières.

[8] https://www.openswan.org/

[9] https://libreswan.org/

[10] https://strongswan.org/

4.1.2 Logiciel OSPF

Pour implémenter le protocole OSPF, seul le logiciel Quagga[11] est disponible sur la plateforme Debian. La communauté est active et le logiciel est régulièrement mis à jour. Le logiciel Quagga est découpé en plusieurs processus ou démons (*daemons* en anglais) qui ont chacun un rôle bien précis :

- zebra : ce processus collecte les informations de routage présent dans le système ainsi que des données issues des autres démons. Il applique les nouvelles routes dans le système.
- bgpd : démon pour le protocole BGP
- ripd et ripngd : démons pour les protocoles RIPv1, RIPv2 et RIPng. RIPng est une implémentation du protocole RIP pour IPV6, standardisé par la norme RFC2080.
- ospfd et ospf6d : ce sont les démons OSPF pour IPv4 et IPv6
- vtysh : ce démon possède un rôle transverse. Il est utilisé pour configurer et interroger l'état des démons précédents.

4.1.3 Logiciel pour les tunnels GRE

Comme nous l'avons vu au paragraphe 3.2.2, OSPF utilise des paquets multicast pour communiquer avec ces voisins. Cependant, comme IPsec n'est pas capable de faire transiter ces paquets, nous ajoutons une couche intermédiaire qui aura pour but de transporter ces informations. Ces tunnels GRE, n'ont pas besoin de logiciel pour fonctionner, car le noyau de Linux intègre toutes les fonctionnalités nécessaires pour les créer. Une fois le paramétrage effectué, nous obtiendrons une nouvelle interface réseau pour chaque nouveau tunnel.

Voici les commandes nécessaires pour établir un tunnel GRE sur un système Linux de type Debian. Dans notre exemple, nous avons deux sites avec les adresses IP publiques suivantes : `1.2.3.4` et `5.6.7.8`. Nous souhaitons créer un tunnel qui possède les adresses IP privées `10.0.0.1` d'un côté et `10.0.0.2` de l'autre.

Sur le premier routeur (`1.2.3.4`), nous lançons les commandes suivantes :

```
ip tunnel add MyGRE mode gre remote 5.6.7.8 local 1.2.3.4 ttl 255
ip link set MyGRE up
ip addr add 172.16.0.1/30 dev MyGRE
```

Voyons en détail ces commandes : sur la première ligne, « *ip tunnel* » crée le tunnel GRE et le nomme « *MyGRE* » en spécifiant l'adresse distante « *5.6.7.8* » et l'adresse locale « 1.2.3.4 ». La seconde ligne active le tunnel. Sur la troisième ligne, nous ajoutons l'adresse IP locale correspondant au tunnel GRE « 172.16.0.1 », c'est-à-dire la porte d'entrée du tunnel ainsi que le sous-réseau auquel il appartient (172.16.0.0 à 172.16.0.3). Nous avons créé la moitié du tunnel, la seconde partie de passe sur le second routeur et les commandes sont symétriques.

Sur le second routeur (`5.6.7.8`) :

```
ip tunnel add MyGRE mode gre remote 1.2.3.4 local 5.6.7.8 ttl 255
ip link set MyGRE up
ip addr add 172.16.0.2/30 dev MyGRE
```

Voilà, notre tunnel GRE opérationnel et il peut déjà faire transiter des données entre les deux sites.

[11] http://www.nongnu.org/quagga/

4.2 Prérequis et paramétrage initial

L'environnement que nous avons choisi pour mettre en œuvre la refonte du système VPN se base sur les outils sélectionnés précédemment, à savoir StrongSwan pour IPsec et Quagga pour OSPF. Dans un premier temps, nous avons choisi d'utiliser les applications avec un paramétrage minimal, c'est à dire en apportant seulement les réglages nécessaires pour vérifier le bon fonctionnement des outils employés.

Les systèmes Linux sont très souvent paramétrés par des fichiers de configuration. Ces fichiers sont éditables avec un simple éditeur de texte. Nous allons étudier dans les pages ci-dessous quelques-uns des éléments majeurs de ces fichiers de configuration portant sur le système VPN.

4.2.1 Configuration de StrongSwan

La configuration d'IPsec avec StrongSwan est répartie sur 2 fichiers : « `ipsec.secrets` » et « `ipsec.conf` ». Le premier fichier stocke les clés nécessaires à la création du tunnel IPsec : l'hôte distant doit avoir une clé identique pour établir le tunnel VPN. Le second fichier comporte la définition des tunnels ainsi que des paramètres choisis pour assurer la sécurité des communications.

Voici en détail la structure du fichier « `ipsec.conf` ». Le premier mot-clé « `conn %default` » indique des paramètres communs qui seront appliqués à tous les tunnels. Ces tunnels seront décrits par la directive « `conn` *`LeNomDuTunnel`* ». Il y aura autant de fois cette directive qu'il y a de tunnels IPsec à créer.

```
#ipsec.conf - StrongSwan IPsec configuration file
conn %default
    authby=secret
    keyexchange=ikev2
    type=transport
    auto=start

# Tunnel VPN entre le serveurA et le serveurB
conn serveurA-serveurB
    leftid=serveurA
    left=1.2.3.4
    rightid=serveurB
    right=5.6.7.8

conn …
```

Détaillons les principaux mots-clés employés dans ce fichier :

- « `authby=secret` » : méthode employée pour l'authentification mutuelle des serveurs IPsec. Nous avons choisi d'utiliser une clé secrète pour sa simplicité de mise en œuvre. Une version plus sécurisée consisterait à mettre en place des signatures électroniques RSA (certificats).
- « `keyexchange=ikev2` » : Protocole d'échange des clés et d'initialisation des tunnels IPsec.
- « `type=transport` » : Type de connexion employée.
- « `auto=start` » : Démarrage automatique du tunnel lors du lancement de StrongSwan.
- « `leftid=serveurA / rightid=serveurB` » : Nom des serveurs dans IPsec uniquement
- « `left=1.2.3.4 / right=5.6.7.8` » : Adresse IP source du serveur A et adresse IP de destination du serveur B entre qui il faut monter le tunnel IPsec.

Sur le serveur B, les commandes seront identiques à l'exception des paramètres `left/leftid` et `right/rightid` qui seront inversées.

La commande « `ipsec statusall` » nous indique le statut de notre premier tunnel :

```
Status of IKE charon daemon (strongSwan 5.2.1, Linux 3.16.0-4-amd64, x86_64):
  uptime: 2 minutes, since Mar 21 21:02:39 2016
Listening IP addresses:
  172.17.1.1
  1.2.3.4
Connections:
   serveurA-serveurB:  1.2.3.4...5.6.7.8  IKEv2
   serveurA-serveurB:   local:  [serveurA] uses pre-shared key authentication
   serveurA-serveurB:   remote: [serveurB] uses pre-shared key authentication
   serveurA-serveurB:   child:  1.2.3.4/32 === 5.6.7.8/32 TRANSPORT
Security Associations (1 up, 0 connecting):
   serveurA-serveurB[1]: ESTABLISHED 2 minutes ago, 1.2.3.4[serveurA]...5.6.7.8[serveurB]
   serveurA-serveurB[1]: IKEv2 SPIs: e5f74b3c8c1089c2_i, pre-shared key reauthentication in 2 hours
   serveurA-serveurB[1]: IKE proposal: AES_CBC_256/HMAC_SHA2_512_256/PRF_HMAC_SHA2_512/MODP_4096
   serveurA-serveurB{1}:  INSTALLED, TRANSPORT, ESP SPIs: c20a8893_i ccb9c7d4_o
   serveurA-serveurB{1}:  AES_CBC_256/HMAC_SHA2_256, 308 bytes (23 pkts, 6s ago), rekeying in 40 minutes
   serveurA-serveurB{1}:   1.2.3.4/32 === 5.6.7.8/32
```

Le tunnel est fonctionnel : la première phase, IKEv2, nous indique un statut « `ESTABLISHED` » en mode transport à l'aide de clés pré-renseignées (« `uses pre-shared key authentication` »). La seconde phase utilise le protocole ESP pour créer le tunnel qui sert à faire transiter les données.

Dans cette première version, nous utilisons la sécurité par défaut pour éviter de rencontrer des problèmes liés à cette configuration. Cette étape est traitée dans le paragraphe 4.3.1.1 détaillant les premiers tests réalisés.

4.2.2 GRE

La définition des tunnels GRE est très proche de la configuration des interfaces réseau traditionnelles sous Debian. Nous avons ajouté un fichier dans le dossier « `/etc/network/interfaces.d/` » portant le nom de « `gre_tunnels.cfg` ». Ce fichier contient tous tunnels à mettre en place. La section présentée ci-dessous sera donc répétée autant de fois qu'il y a de tunnels GRE à créer sur ce serveur.

```
#Fichier de configuration des tunnels GRE
auto serveurAserveurB
iface serveurAserveurB inet tunnel
    mode gre
    address 172.16.0.1
    netmask 255.255.255.252
    endpoint 5.6.7.8
    local 1.2.3.4
```

- « `address 172.16.0.1` » : Adresse IP qui est attribuée à la nouvelle interface GRE. Cette interface est le point d'entrée du tunnel.
- « `netmask 255.255.255.252` » : Masque de sous-réseau. Ce masque définit le nombre d'adresses IP qui seront contenues dans le sous-réseau. Ici nous avons opté pour la taille minimale c'est-à-dire 2 adresses IP utilisables. La première adresse est attribuée au serveur A et la seconde au serveur B.
- « `endpoint 5.6.7.8` » : adresse IP publique du serveur destinataire du tunnel GRE.
- « `local 1.2.3.4` » : Adresse IP publique du serveur local d'où partira le tunnel GRE.

Ces commandes sont un peu différentes que celles évoquées dans *4.1.3 - Logiciel pour les tunnels GRE*, car elles ne s'appliquent pas dans le même contexte : les commandes ci-dessus sont spécifiques

au fichier de configuration des interfaces réseaux tandis que les précédentes s'utilisent directement dans une console Linux de type BASH.

4.2.3 OSPF

Quagga est composé de plusieurs processus (cf. *4.1.2 - Logiciel OSPF*). Chacun de ces processus est configuré avec un fichier qui lui est propre. Pour notre système VPN, nous n'avons besoin que des processus Zebra et OSPF.

Le premier fichier « `/etc/quagga/ospfd.conf` » décrit le routeur OSPF et plus particulièrement ces interfaces réseaux comme nous pouvons le voir ci-dessous :

```
router ospf
    ospf router-id 1.2.3.4
    redistribute static
    passive-interface lo
    passive-interface eth1
    passive-interface eth2
    network 192.168.0.0/24 area 0.0.0.0
    network 192.168.50.0/24 area 0.0.0.0
    network 172.16.0.1/30 area 0.0.0.0
```

- « `ospf router-id 1.2.3.4` » : Chaque routeur possède un nom unique dans le réseau OSPF. Ce nom se présente sous la forme d'une adresse IP. Par habitude, nous utilisons l'adresse IP publique du routeur comme identifiant.
- « `redistribute static` » : Cette directive est très importante, elle a pour but de transférer les règles qui sont échangées avec OSPF à la machine Linux hôte. Si ce paramètre est omis, OSPF possèdera bien une représentation complète du réseau, mais ne la partagera pas avec le système d'exploitation ce qui enlève tout l'intérêt de mettre en place cet outil.
- « `passive-interface lo, eth1,…` » : Aucun échange OSPF n'est effectué sur ces interfaces réseaux.
- « `network 192.168.0.0/24 area 0.0.0.0` » : Réseau connecté directement au routeur à transmettre aux autres équipements OSPF.
- « `network 172.16.0.1/30 area 0.0.0.0` » : Les tunnels GRE sont aussi à inscrire, car ils sont nécessaires pour que le routeur puisse communiquer avec ses congénères.

Le second fichier « `/etc/quagga/zebra.conf` » est chargé de décrire les réseaux qui ne sont pas reliés directement au routeur pour que celui-ci les transmette via OSPF. La directive à employer dans ce fichier est très simple : « `ip route 192.168.111.0/24 192.168.2.1` ». Cette commande correspond à la déclaration du sous-réseau `192.168.111.0/24`. Elle indique que pour atteindre ces machines, il faut envoyer les trames réseau reçues à la machine ayant pour IP `192.168.2.1`.

Nous pouvons prendre comme exemple une entreprise qui possède plusieurs bâtiments sur un même site, et qui a installé différents routeurs secondaires dans chacun d'entre eux. Cette adresse IP correspondrait à un de ces routeurs ; les machines installées dans ce bâtiment forment le sous-réseau en question.

4.3 Mise en place de plateformes de tests

4.3.1 Tests avec des machines virtuelles

4.3.1.1 Plateforme de test basique

La première plateforme de test avait pour but de tester toute la chaine IPsec-GRE-Quagga en commençant par IPsec.

Les machines Debian n'ont pas besoin de beaucoup de puissance pour fonctionner (512Mo de RAM, et 8 Go d'espace disque par machine). J'ai donc installé sur ma machine professionnelle un environnement complet simulant trois agences et une quatrième qui représente Internet. Cet environnement est réalisé avec des outils de virtualisation standard, dans mon cas Microsoft Hyper-V.

Nous avons choisi de créer une machine « Internet01 » pour avoir la possibilité « d'écouter » le trafic qui y transite. De plus, nous avons appliqué des restrictions réseaux sur cette machine afin de ne laisser passer que le trafic nécessaire au VPN, c'est-à-dire les protocoles IKE et ESP utilisées par IPsec.

La *Figure 20* décrit l'infrastructure mise en place lors de ces tests. Les 3 machines virtuelles sur la gauche représentent les routeurs des agences de Netapsys, à savoir Rhône-Alpes (*IPsec01-ra*), Paris-rue de Provence (*IPsec01-PR*) et Paris-rue de Mogador (*IPsec01-mo*). Chaque routeur possède deux interfaces réseau : la première vers le réseau local, c'est-à-dire les postes des utilisateurs ou les serveurs, et la seconde vers Internet. La machine virtuelle de droite fournit l'accès Internet et autorise le transfert des données entre les routeurs à l'aide de règles de routage écrites avec *iptables*.

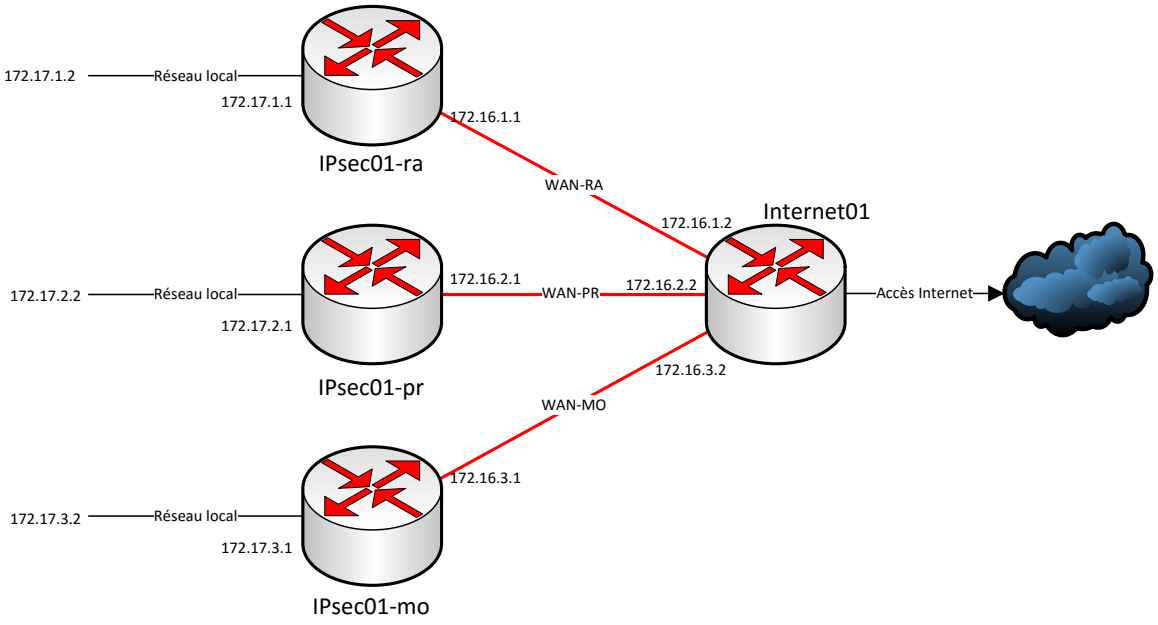


Figure 20 : Premier environnement de test avec des machines virtuelles

Les premiers résultats se sont révélés concluants malgré une configuration complexe à mettre en place. La propagation des règles de routage des différents sites à l'aide de Quagga a validé ce premier test.

Pour améliorer la sécurité des échanges, nous avons mis en place la sécurité des échanges IPsec à l'aide des protocoles IKE et ESP. Pour cela, nous avons ajouté les lignes suivantes dans le fichier « `ipsec.conf` » dans la section « `conn %default` ».

```
ike=aes256-sha512-modp4096!
esp=aes256-sha512- modp4096!
```

Nous pouvons voir que le paramètre utilisé pour configurer IKE et ESP, « `aes256-sha512-modp2048` », est composé de 3 parties. Voici à quoi correspondent ces valeurs :

- AES256 nous indique l'algorithme de chiffrement utilisé. AES est développé par le l'institut National des Standards et Technologies (*National Institute of Standards and Technology - NIST*) et fournit un algorithme sécurisé et efficace pour les données sensibles d'après [Cameron & Woodberg, 2013]. AES est disponible avec différentes longueurs de clé, le plus souvent 128, 256 et 384. En augmentant la longueur de la clé, on augmente la sécurité des échanges. Cependant, la puissance nécessaire pour chiffrer et déchiffrer les données augmente de la même manière.
- SHA512 est l'algorithme utilisé pour assurer l'intégrité des données. Les algorithmes SHA ont pour but de créer une empreinte d'un message en produisant un résultat de taille fixe. Cette empreinte est unique pour un message donné.
- Enfin, modp4096 est lié à la longueur des clés utilisées par l'algorithme de *Diffie-Hellman*.

Nous avons choisi ces paramètres, car d'après *l'ANSSI*, ils sont suffisants pour établir des échanges sécurisés entre les participants [Agence nationale de la sécurité des systèmes d'information (ANSSI), 2015]. Toutefois, les valeurs utilisées actuellement sont facilement modifiables et suivront les évolutions futures grâce à la flexibilité du protocole IKE.

Notre première architecture VPN IPsec est représentée par la *Figure 21*. Nous obtenons une communication en triangle avec un partage des données de routage. En effet, chaque routeur transmet, via OSPF, à ses congénères la liste des réseaux auquel il est relié.

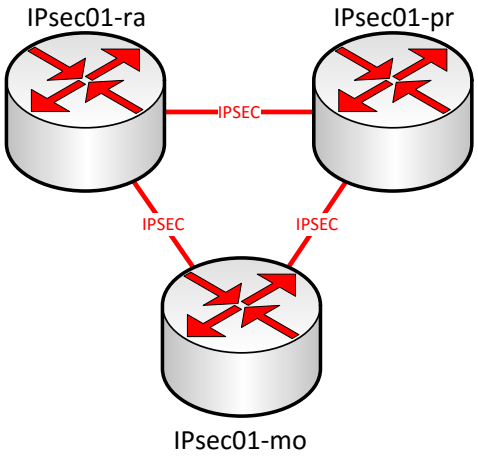


Figure 21 : Représentation logique des tunnels IPsec du premier test

4.3.1.2 Plateforme de test avancée

Nous avons souhaité faire évoluer cette plateforme en ajoutant une seconde connexion Internet aux routeurs et ainsi créer un VPN entièrement maillé (*Figure 22*). Nous souhaitions vérifier l'efficacité du routage dynamique dans le cadre d'un changement de topologie. Cette évolution nous a amenés à ajouter des priorités sur les tunnels IPsec gérés par OSPF.

Ces priorités sont définies dans le fichier « `ospfd.conf` » de Quagga. Pour cela, il faut ajouter les lignes suivantes pour chaque lien GRE que nous avons créé.

```
interface serveurAServeurB
    ip ospf cost 50
```

Le cout, représenté par le chiffre 50 dans notre exemple, est à mettre en lien avec les couts des autres liens déclarés pour les autres tunnels GRE. Le cout le plus faible donnera la priorité sur un lien plutôt que sur les autres liens disposant d'un cout supérieur. Une bonne pratique consiste à paramétrer les couts en fonction de leur débit Internet. Si nous avons deux liens partant du même routeur avec des débits de 100Mbits et 200Mbits, nous pouvons indiquer par exemple un cout de 60 pour le premier et 30 pour le second. Cette fonctionnalité est très utilisée dans le cadre de topologies maillées pour faire transiter le trafic VPN par certains routeurs spécifiques du réseau.

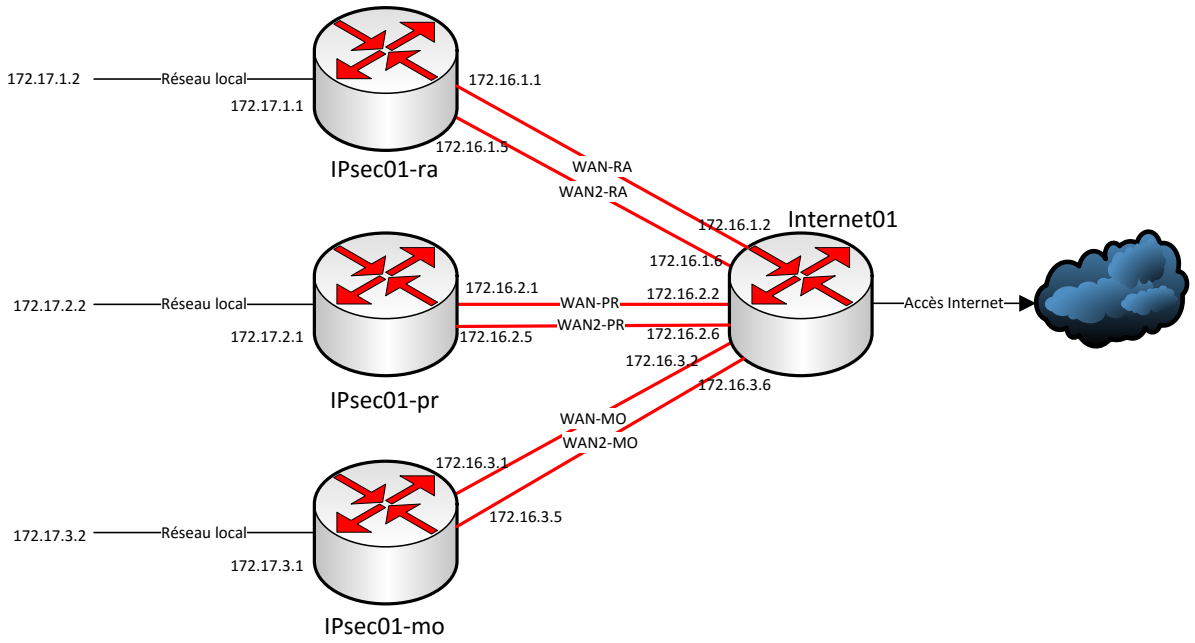


Figure 22 : Test avec des liens Internet doublés

Comme vous pouvez le constater sur la *Figure 23*, la complexité de notre nouveau test s'est considérablement accrue. Nous sommes passés de 3 tunnels IPsec et 3 tunnels GRE, à 12 tunnels de chaque type. L'industrialisation des paramètres de chaque logiciel est devenue nécessaire. Notre outil de gestion des configurations, CFEngine, est justement prévu pour réaliser ce type d'opérations. Nous évoquerons son fonctionnement dans le paragraphe 4.4.

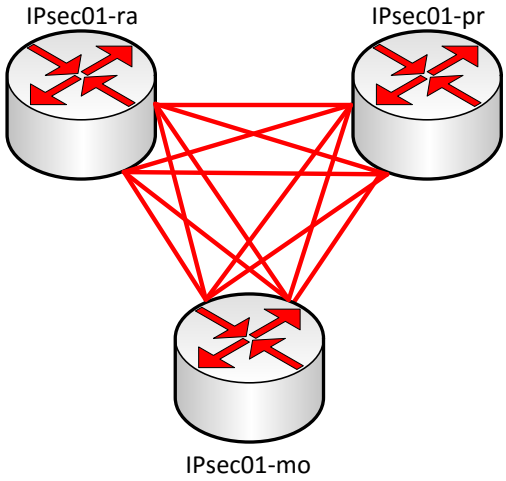


Figure 23 : Représentation logique des tunnels IPsec du second test

4.3.2 Première architecture multisite

Suite à ces différents tests, nous avons opté pour une configuration plus proche de la réalité. Nous disposons de quelques serveurs inutilisés sur une partie de nos sites qui ont servi à réaliser une architecture VPN IPsec en parallèle du VPN utilisé en production.

Nous avons donc paramétré 3 serveurs sur les sites de Lyon, Paris-rue de Provence et Paris-rue de Mogador. Le 4e site, OVH, étant dépourvu de machines physiques, nous avons opté pour une nouvelle machine virtuelle identique à celle utilisée en production actuellement. Ces 4 sites étaient tous connectés à Internet avec différents opérateurs, comme c'est le cas dans notre environnement de production. Hormis quelques problèmes de routage initiaux, notre nouveau VPN IPsec s'est comporté normalement et nous avons pu tester la fiabilité du système. En coupant un des tunnels IPsec sur un routeur, OSPF recalcule automatiquement la nouvelle table de routage et la transmet aux autres équipements. Malgré cette coupure de quelques secondes, nous pouvions toujours contacter le site en question, mais notre trafic était relayé par un autre routeur ce qui se traduit par des temps d'accès légèrement plus lents.

Cette infrastructure dans des conditions proches de la réalité a permis d'affiner les processus d'installation des différents composants et de valider les configurations des différents outils employés. Nous avons aussi défini quelques paramètres supplémentaires portant sur les temps de détection des liens défaillants pour accélérer le processus de reconfiguration.

4.4 Industrialisation des configurations

CFEngine ne possède pas de commandes natives pour piloter les logiciels que nous avons sélectionnés. Cependant, il est capable de déposer et d'exécuter des scripts que nous avons préparés. Le langage le plus courant sous les environnements Linux est le BASH (*Bourne-Again Shell*). C'est un langage de script simple à prendre en main et qui ne nécessite pas de compilation pour être exécuté.

4.4.1 Données sources

Nous avons créé un script BASH pour chaque fichier de configuration à paramétrer ainsi qu'un script supplémentaire qui est chargé de piloter le tout. Les données nécessaires au paramétrage sont

stockées dans un fichier JSON (*JavaScript Object Notation*). Le fichier JSON est issu de la notation des objets du langage JavaScript. Il permet de représenter de l'information de manière structurée, à la manière du XML.

Voyons en détail la structure du fichier JSON, « `vpn_serveur.json` », que nous avons créé:

```
{
   "ipsecsrv": [
        {
            "hostname" : "routeur01.lyon.netapsys.fr",
            "wan" : [
                {
                    "description" : "Fibre Orange",
                    "ip" : "80.80.80.80",
                    "subnet" : "80.80.80.80/32",
                    "name" : "ra01",
                    "cost" : "10"
                },
                     {...}
            ],
            "lan" : [
                {
                    "description" : "LAN des utilisateurs",
                    "group" : "subnet_utilisateurs",
                    "name" : "eth2",
                    "subnet" : "192.168.88.0/24"
                },
                     {...}
            ],
            "lanindirect" : [
                {
                    "description" : "sous-réseau client A",
                    "group" : "acces_client",
                    "subnet" : "192.168.24.240/30",
                    "gateway" : "192.168.1.14"
                },
                     {...}
            ]
        },
        {...}
    ]
}
```

Cet extrait relatif au routeur de l'agence de Lyon est composé de 3 parties : la première représente la connexion Internet avec son IP publique et le cout de cette connexion. La seconde section décrit les sous-réseaux directement reliés au routeur. Enfin, la dernière partie détaille les sous-réseaux situés sur le site de Lyon, mais pour lesquels il faut s'adresser à un intermédiaire pour les contacter.

La première section sert à créer les tunnels IPsec et GRE, la seconde et la dernière sont intégrés à OSPF qui se charge de les diffuser aux autres routeurs de l'entreprise. Pour extraire les données de ce fichier, nous utilisons la commande Linux « `jq` ». Par exemple, pour récupérer le nom des connexions Internet de d'un routeur, la commande est la suivante :

```
cat $json | jq '.ipsecsrv[] | select(.hostname == $host) | .wan[].name'  --arg host $1
```

Cette commande à besoin du paramètre `$1` qui correspond au nom du routeur sur lequel je lance la commande. Le résultat de cette requête est `"ra01"`.

Nous avons créé une petite bibliothèque de fonctions supplémentaires pour extraire les données du fichier json. De cette manière, nous pouvons faire évoluer la structure de ce fichier librement en ne mettant à jour que les fonctions associées. Les autres scripts ne seront pas impactés par ces modifications.

4.4.2 Tunnels GRE

Le fichier suivant se rapporte à la configuration des tunnels GRE. La configuration finale est le résultat d'une boucle sur tous les tunnels qui sont contenus dans la variable « `$MyTunnels` ». Pour chaque résultat, les données sont éclatées dans différentes variables qui sont à leur tour écrites dans le fichier final. La fonction « `GetWanIpByWanName` » située en fin de script renvoie les données issues du fichier json évoqué précédemment. La variable « `$MyTunnels` » est aussi le résultat d'une fonction que nous avons développé. Le script principal lance la fonction une fois et conserve le résultat en mémoire pour les différentes utilisations.

```
#!/bin/bash
# Génération de la config des interfaces GRE
echo "#Fichier généré par CFEngine : ne pas modifier !"
for tunnel in $MyTunnels ; do
    IFS="-" read sideA sideB subnet sideAip sideBip <<< $tunnel
    echo "auto $sideA$sideB
iface $sideA$sideB inet tunnel
    mode gre
    address $sideAip
    netmask 255.255.255.252
    endpoint $(GetWanIpByWanName $sideB)
    local $(GetWanIpByWanName $sideA)
    "
done
```

4.4.3 Script pour OSPF

L'extrait suivant concerne la mise à jour des fichiers de configuration d'OSPF. Nous utilisons un script principal, qui est chargé de générer ces fichiers puis de relancer les services s'il y a eu une modification. Nous commençons donc par générer le fichier dans un dossier temporaire avec la commande « `source vpn_create_quagga_ospfd.sh > "${FichierSource}"` ». Ensuite, nous appelons la fonction « `CompareAndCopy` » qui se chargera de comparer le fichier utilisé actuellement et celui que nous avons créé. Cette fonction mettre à jour le fichier en question s'ils sont différents. Enfin, nous mettons en place un *drapeau*, une sorte de post-it, pour indiquer qu'il faut relancer le service gérant OSPF : Quagga. Ce drapeau est vérifié à la fin du fichier et le service redémarré si nécessaire.

```
…
# Fichiers de config Quagga : OSPF
FichierSource="$tmpout/ospfd.conf"
FichierDest="/etc/quagga/ospfd.conf"
#Création du fichier
source vpn_create_quagga_ospfd.sh > "${FichierSource}"
#est-ce que les fichiers sont identiques ?
result=$(CompareAndCopy "${FichierSource}" "${FichierDest}")
#est-ce que je dois recharger ?
if [ $result -eq 1 ] ; then
    FlagRestartQuagga="true"
fi
…
```

4.4.4 Attribution d'adresses IP GRE

Le dernier extrait se rapporte à la génération des tunnels GRE et plus spécifiquement à l'attribution des adresses IP pour chaque tunnel.

Dans notre infrastructure, nous avons les tunnels IPsec, basés sur les adresses IP publiques des liens Internet. Sur ces tunnels IPsec, les tunnels GRE sont créés et ajoutent le support du multicast entre nos sites. Les données des tunnels GRE étant encapsulées dans les trames IPsec, les informations sont sécurisées. Cependant, chaque tunnel GRE crée une interface virtuelle de chaque côté du tunnel, c'est-à-dire une carte réseau fictive à laquelle il faut attribuer une adresse IP. Nous avons besoin de séparer les réseaux de chaque tunnel GRE en utilisant des sous-réseaux distincts. Ces sous-réseaux n'ayant besoin que de 2 adresses IP chacun, une pour chaque hôte du tunnel GRE, nous allons donc utiliser un masque de sous-réseau en 255.255.255.252.

Le premier sous-réseau ainsi créé, avec l'adresse 172.16.0.0/30, est formé de la manière suivante :

Tableau 6 : Détail du premier sous-réseau pour les tunnels GRE

Adresse	Description
172.16.0.0	Première adresse IP = adresse du sous-réseau
172.16.0.1	Adresse IP attribuée au serveur A
172.16.0.2	Adresse IP attribuée au serveur B
172.16.0.3	Dernière adresse IP = adresse de broadcast (non utilisé)

Le second sous-réseau utilise les adresses suivantes : 172.16.0.4/30, le troisième : 172.16.0.8/30, etc. Cette numérotation interne des interfaces GRE est très importante, car si deux hôtes se trouvent par erreur avec des adresses IP dans des sous-réseaux différents, ils ne pourront pas échanger de données. Pour éviter ce problème, nous avons créé plusieurs fonctions BASH qui s'exécutent sur les hôtes et qui sont chargées de générer ces adresses IP. Ces fonctions sont identiques pour tous les serveurs VPN et sont envoyées par CFEngine.

La première fonction, « `GenerateAllTunnelsGRE` », dont le détail se trouve en (*Annexe VII*), renvoi la liste complète des tunnels possibles pour notre infrastructure d'après le fichier « `vpn_serveur.json` ». Cette liste est ensuite filtrée par la fonction *GenerateMyTunnelsGre* (*Annexe VIII*) qui ne conserve que la liste des tunnels à paramétrer sur le serveur où ces fonctions sont exécutées. Les données renvoyées sont celles que nous retrouvons dans la partie *4.4.2 - Tunnels GRE* dans la variable « `$MyTunnels` ».

Pour générer et attribuer les adresses IP des tunnels GRE, la fonction précédente fait appel à la méthode qui se nomme « `GetGreIP` ». Cette sous-fonction est très intéressante, nous pouvons voir son fonctionnement ci-dessous :

```
# Fonction permettant de calculer les adresses IP des sous-réseaux GRE
# $1 = N° du tunnel
# $2 = Adresse IP à renvoyer : Local, Remote, Subnet ?
function GetGreIP () {

    # calcul de l'adresse IP sous la forme A.B.C.D sachant que A=172 et B=16
    # D = N°Tunnel * 4 (car il y a 4 IP par tunnel) -3, car le tunnel n°1 = IP .1
    GreIP_D=$(($1 * 4) - 3)

    # Calcul de C
    # Si N°tunnel > 64 alors il faut incrémenter C
    if [[ "$1" > "64" ]] ; then
        GreIP_C=$($GreIP_D / 256)
    else
        # on initialise C
        GreIP_C=0
    fi

    # si N°tunnel > 64 alors D > 256, ce qui n'est pas possible dans une IPv4
    # donc utilisation du modulo 256
    GreIP_D=$(($GreIP_D % 256))

    if [[ "$2" == "Local" ]] ; then
        echo "172.16.$GreIP_C.$GreIP_D"
    fi

    if [[ "$2" == "Remote" ]] ; then
        ((GreIP_D++))
        echo "$172.16.$GreIP_C.$GreIP_D"
    fi

    if [[ "$2" == "Subnet" ]] ; then
        ((GreIP_D--))
        echo "$172.16.$GreIP_C.$GreIP_D"
    fi
}
```

Dans le détail, la fonction « `GetGreIP` » récupère le numéro du tunnel fourni par la méthode « `GenerateAllTunnelsGRE` ». Ce numéro est généré tout simplement en numérotant les tunnels lors de leur création. Avec ce système, le numéro est identique sur chacun des serveurs qui hébergeront les outils VPN, la cohérence des adresses IP est ainsi respectée.

Ce numéro est multiplié par 4, car nos sous-réseaux ont besoin de 4 adresses IP chacun. Si nous obtenons plus de 64 tunnels GRE, nous incrémentons le troisième chiffre de l'adresse IP : 172.16.**1**.0 pour obtenir une adresse IP cohérente.

4.5 Routage dynamique et pare-feu

Avec le routage dynamique, nous nous sommes rendu compte que tous les sous-réseaux étaient accessibles à partir de n'importe quel site. Cette largesse était déjà en place avec OpenVPN dans une moindre mesure. Cependant, nous avons décidé de traiter cet aspect en utilisant le pare-feu installé sur nos machines-routeurs pour distiller les autorisations d'accès que nous souhaitons mettre en place.

Dans les systèmes Linux, le pare-feu, intégré au noyau, est piloté par le logiciel *iptables*[12]. Le pare-feu a pour mission de protéger la machine des intrusions, mais aussi d'autoriser (ou d'interdire) les paquets réseau à emprunter un chemin particulier.

Nous avons couplé au système VPN une gestion des accès aux sous-réseaux en utilisant les données du fichier JSON. Pour chaque sous-réseau déclaré dans ce fichier, nous lui avons attaché un groupe de référence. Ces groupes sont utilisés comme dans une table de vérité pour créer les autorisations d'accès souhaitées.

Prenons l'exemple de trois groupes de sous-réseaux : un groupe « `pc_utilisateurs` », un groupe « `serveurs_recette` » et un dernier groupe « `wifi_clients` ». Avec notre système, nous créons les règles suivantes :

```
iptables -A FORWARD -m set --match-set pc_utilsateurs src -m set --match-set serveurs_recette dst -j ACCEPT
iptables -A FORWARD -m set --match-set pc_utilsateurs src -m set --match-set wifi_clients dst -j REJECT
iptables -A FORWARD -m set --match-set serveurs_recette src -m set --match-set pc_utilsateurs dst -j REJECT
iptables -A FORWARD -m set --match-set serveurs_recette src -m set --match-set wifi_clients dst -j REJECT
iptables -A FORWARD -m set --match-set wifi_clients src -m set --match-set pc_utilsateurs dst -j REJECT
iptables -A FORWARD -m set --match-set wifi_clients src -m set --match-set serveurs_recette dst -j ACCEPT
```

Ces règles sont envoyées au logiciel iptables qui les appliquera pour décider si une trame réseau doit être transmise ou au contraire rejetée. Chaque règle de l'exemple ci-dessus fonctionne de la même manière : nous avons un groupe source et un groupe destination. Si les données à transférer correspondent à la règle, l'action qui se trouve en fin de ligne est appliquée (ACCEPT/ REJECT). Les mots-clés « `-m set –match-set` » indique à iptables qu'il doit traiter le bloc comme un groupe et pas comme une adresse unique. Dans cet exemple nous avons autorisé ou interdit tout le trafic, mais il est possible de limiter l'autorisation à seulement un protocole (ICMP, TCP, UDP…) et même à un seul port pour être encore plus précis (TCP/80 pour le trafic HTTP, TCP/22 pour l'accès SSH, …)

Notre script (*Annexe IX*) se charge de créer les groupes et de les alimenter avec un algorithme assez simple de parcours de fichier. Nous avons aussi introduit une notion de groupe personnalisé. Ces groupes sont déclarés dans un autre fichier JSON, « `iptables_groups.json` », de la manière suivante :

```
{
    "groups": [
        {
            "name" : "Serveurs_DNS" ,
            "subnets" : [
                { "description" : "ns01.mo",  "subnet" : "192.168.10.9/32" },
                { "description" : "ns01.ra",  "subnet" : "192.168.14.9/32" },
                { "description" : "ns01.ovh", "subnet" : "192.168.42.9/32" },
                { "description" : "ns02.pr",  "subnet" : "192.168.90.9/32" }
            ]
        },{
            "name" : "Serveurs_cfengine" ,
            "subnets" : [
                { "description" : "cf01.mo",  "subnet" : "192.168.10.15/32" },
                { "description" : "cf01.ovh", "subnet" : "192.168.42.16/32" }
                      ]
        }
       ]
    }
}
```

Dans cet extrait nous définissons deux groupes de serveurs ainsi que leurs membres : les serveurs DNS et les serveurs CFEngine. Ces deux groupes sont exploitables de la même manière que les groupes définis dans le fichier « `vpn_serveur.json` » contenant les données des routeurs à paramétrer. Il est bien sûr possible de les mixer entre eux pour créer des règles telles que : « j'autorise les membres du groupe des « `pc_utilisateurs` » à se connecter aux serveurs du groupe « `serveurs_DNS` » sur le port TCP/53 ».

[12] http://www.netfilter.org/

Au final, notre système est très flexible, car cette gestion de groupe simplifie grandement la tâche des administrateurs système. Nous avons une quinzaine de groupes au total ce qui nous donne une table de vérité de seulement 200 lignes d'autorisations. Créer ces règles à la main sans gestion de groupe aurait été une source d'erreur et de dysfonctionnement permanent et aurait nécessité plusieurs milliers de lignes.

4.6 Difficultés rencontrées

Durant les différentes phases de test, nous avons rencontré plusieurs difficultés techniques qui ont ralenti le projet, mais que nous avons surmontées avec succès.

La première épreuve à surmonter était liée aux tunnels GRE. Nous avons tenté par différentes méthodes d'obtenir un environnement globalisé capable de faire transiter les trames multicast nécessaire pour OSPF, sans pour autant arriver à un résultat concluent. La version 5.3[13] de StrongSwan est, *à priori*, capable de faire transiter ce type d'informations. Nous avons donc installé cette version sur notre environnement de test avec les machines virtuelles. Malgré plusieurs tentatives, nous n'avons pas obtenu de résultats satisfaisants. Cette fonctionnalité, très intéressante, aurait pu supprimer la complexité inhérente à la gestion des tunnels GRE.

Nous avons aussi essayé de créer un tunnel GRE multipoint incluant tous les sites de l'entreprise. En quelque sorte, il s'agissait d'obtenir un commutateur virtuel reliant tous les sites avec une seule interface GRE par serveur. Ce commutateur virtuel avait pour avantage de laisser passer les trames multicast et donc d'autoriser la configuration d'OSPF assez simplement. Cette technologie a déjà fonctionné par le passé avec des logiciels plus anciens et quelques composants additionnels[14]. Cependant, nous ne sommes pas arrivés à faire fonctionner ce tunnel GRE multipoint avec StrongSwan.

Enfin le dernier point noir que nous avons corrigé concerne la taille des trames qui circulent sur Internet. Nous nous sommes heurtés à un problème de perte d'informations : certains paquets n'arrivaient pas à destination, car ils étaient supprimés par des routeurs situés sur Internet. La taille par défaut est de 1500 bytes, à laquelle il faut enlever les informations utilisées par IPsec et par GRE. Dans ce cas, les données transitent normalement. L'opérateur Internet de l'agence de Lyon, Orange, fournit une connexion avec un MTU (*Maximum Transmission Unit*) de seulement 1458bytes au lieu des 1500 en général. Le MTU étant la taille maximale d'un paquet pouvant être transmis en une seule fois sur une interface. Dans ce cas, les paquets qui dépassaient cette valeur étaient supprimés. Après plusieurs essais, nous avons dû limiter la taille des paquets envoyés sur Internet en forçant le tunnel GRE à faire des trames plus petites (1342 bytes).

Au final, la nouvelle plateforme que nous avons testée dans différents scénarios s'avère pleinement fonctionnelle. L'industrialisation, opérée avec CFEngine et les scripts BASH créés pour l'occasion, simplifie et automatisent la configuration et la maintenance du nouveau VPN. Il reste cependant à réaliser l'étape de mise en production en ne perturbant pas le travail des collaborateurs de l'entreprise.

[13] https://strongswan.org/blog/2015/03/30/strongswan-5.3.0-released.html

[14] https://sourceforge.net/p/opennhrp/support-requests/3/#df11

Chapitre 5

Déploiement du nouveau système

5.1 Prérequis au déploiement

5.1.1 État des lieux

Nos serveurs utilisés actuellement comme routeur, pare-feu et système VPN possèdent des systèmes d'exploitation différents : nous utilisons deux distributions Linux distinctes, à savoir *Debian* et *CentOS*. Les versions de *Debian* sont aussi disparates : il y a des Debian « Wheezy » (version 7) et Debian « Jessie » (version 8) comme nous le montre le *Tableau 7*.

Tableau 7 : Inventaire des pare-feu de Netapsys

Site	Nom interne	Marque / Modèle	Système d'exploitation	Version	Nombres de ports réseau
Lille	fw01.no	Lanner FW-7573	Linux Debian	8.2	6
Lyon	fw02.ra	Dell R410	Linux Debian	7.9	6
Madagascar	fw01.mg	Dell 2850	Linux CentOS	6.7	4
Nantes	fw01.at	Dell 2950	Linux Debian	7.9	6
OVH	fw01.ovh	Machine virtuelle	Linux Debian	7.8	2
Paris - Mogador	fw02.mo	Dell R300	Linux Debian	7.8	6
Paris - Provence	fw02.pr	Dell 1950	Linux Debian	7.9	6
Rennes	fw01.bz	Zotac ZBOX-ID42-BE	Linux Debian	8.2	2
Strasbourg	fw01.ge	PC Asus	Linux Debian	7.9	6

Nous avons choisi d'utiliser Debian 8 pour le système VPN pour les raisons suivantes :

- L'équipe « Infra » possède une bonne connaissance de cette distribution Linux. Elle est majoritairement utilisée chez Netapsys.
- Nous souhaitons uniformiser nos systèmes pour obtenir un comportement global cohérent.
- Les versions des logiciels utilisés pour le VPN, à savoir StrongSwan et Quagga, sont assez différentes entre les versions de Debian.

En effet, Debian est toujours disponible en trois versions (trois branches) qui sont :

- *Stable* : version figée où les seules mises à jour sont des correctifs de sécurité.
- *Testing* : future version stable où seuls les paquets suffisamment matures peuvent rentrer.
- *Unstable* : surnommée *Sid*, il s'agit d'une version en constante évolution, alimentée sans fin par de nouveaux paquets ou de mises à jour de paquets déjà existants.

La version 8 de Debian correspond à la version stable actuelle et propose StrongSwan en version 5.2 alors que la version 7, « Wheezy », inclut seulement la version 4.5 du logiciel.

Cette différence est importante, car beaucoup de nouveautés sont apparues avec StrongSwan version 5[15].

5.1.2 Uniformisation des systèmes d'exploitation

Pour déployer le VPN et le routage dynamique, nous avons besoin d'uniformiser nos pare-feu. Dans certaines agences, c'est le serveur de test qui est promu en serveur de production. Cette manipulation possède des avantages indéniables, car nous pouvons préparer l'environnement et minimiser la coupure réseau lors de la bascule. De plus, avec l'ancien serveur toujours disponible, une opération dite de « Roll-back » ou marche arrière en français est toujours possible.

Tableau 8 : Pare-feu uniformisés

Site	Nom interne	Marque / Modèle	Système d'exploitation	Version	Nb de ports réseau	Action d'uniformisation
Lille	fw01.no	Lanner FW-7573	Linux Debian	8.4	6	Debian mis à jour
Lyon	fw01.ra	Lanner FW-7573	Linux Debian	8.4	6	Remplacé par un Lanner
Madagascar	fw02.mg	Lanner FW-7573	Linux Debian	8.4	6	Remplacé par un Lanner
Nantes	fw02.at	Lanner FW-7573	Linux Debian	8.4	6	Remplacé par un Lanner
OVH	fw02.ovh	Machine virtuelle	Linux Debian	8.4	3	Création d'une nouvelle VM
Paris - Mogador	fw01.mo	Dell R200	Linux Debian	8.4	6	Ajout de cartes réseau
Paris - Provence	fw01.pr	Dell R710	Linux Debian	8.4	6	Ajout de cartes réseau
Rennes	fw02.bz	Lanner FW-7573	Linux Debian	8.4	6	Debian mis à jour
Strasbourg	fw02.ge	Lanner FW-7573	Linux Debian	8.4	6	Remplacé par un Lanner

Ce projet coïncide aussi avec la mise en place de pare-feu de marque *Lanner*[16]. Ces équipements sont installés, dans un premier temps, sur les agences que nous avons créées récemment (Lille, Rennes, Strasbourg). Vu que ces sites ne possèdent pas d'autre serveur physique, nous avons choisi ces équipements à la fois silencieux et performant, tout en conservant un encombrement minimum.

5.2 Réalisations pratiques

5.2.1 Activation du premier tunnel VPN

Le premier tunnel IPsec est entré en fonction mi-avril entre les sites de Lyon et de Madagascar. Ces sites représentaient de bons candidats pour une première migration grâce à leur position géographique : Mihamina, un des membres de l'équipe infra de l'agence de Madagascar pouvait réaliser l'opération sur place pendant que je réalisais les mêmes manipulations sur le site de Lyon. L'opération a duré une heure environ et s'est effectuée après les horaires habituels de travail pour éviter d'importuner les développeurs.

[15] https://www.strongswan.org/blog/2012/06/20/bye-bye-pluto.html

[16] http://www.lannerinc.com/products/x86-network-appliances/x86-rackmount-appliances/fw-7573

Le travail de migration a débuté par une phase de préparation que nous avons effectuée en amont sur un équipement secondaire, comme indiqué dans le *Tableau 9*. Cet équipement est devenu le pare-feu principal pour ces agences après l'étape de migration.

Le *Tableau 9* détaille les étapes majeures de la préparation :

Tableau 9 : Préparation à la migration

N°	Titre de l'étape	Détail
1	Installation de Debian 8	Installation classique, avec seulement un serveur SSH en nommant la machine avec un nom temporaire (srv01 par exemple)
2	Installation de CFEngine	Installation comme dans toutes les machines Linux de Netapsys
3	Configuration du réseau	Paramétrage de chaque interface physique qui est utilisée dans le VPN
4	Activation du Relay DHCP	Fonction importante, car sinon nos machines et nos utilisateurs ne se connecteront pas au réseau
5	Installation et configuration d'OpenVPN	L'ancien VPN est utilisé tant que Paris-rue de Provence n'est pas migré vers le nouveau système
6	Configuration du futur VPN	Les fichiers *json* décrivant le VPN IPsec sont mis à jour et débarrassés de toutes les données utilisées pour les tests
7	Préparation des règles de routage locales	Copie des réglages actuels, puis modification pour coller avec le futur pare-feu de chaque site, notamment sur la dénomination et le nombre d'interfaces disponibles
8	Préparation des règles OpenVPN	Le trafic entre les sites ne doit plus être déclaré dans OpenVPN. Nous préparons une configuration en enlevant ces routes

Quelques précisions sont nécessaires notamment sur l'étape 1 : l'utilisation du nom définitif de la machine aurait pour conséquence d'installer toutes les fonctionnalités du futur pare-feu via CFEngine. En effet, les fonctionnalités « pare-feu » sont installées automatiquement si le nom de la machine contient « fw ». C'est une méthode de paramétrage des machines avec CFEngine qui est très utile au quotidien, car juste avec le nom, nous déployons automatiquement les fonctionnalités associées. Nous utilisons cette technique pour paramétrer les serveurs DNS (« ns » dans le nom), LDAP ou encore CFEngine (« cf » dans le nom). Si nous avions défini le futur nom de la machine dès à présent, nous aurions obtenu un serveur totalement muet, car celui-ci aurait été en conflit avec le serveur existant sur le réseau.

Dans l'étape 5, nous configurons l'ancien VPN sur la machine pour que les sites de Lyon et de Madagascar puissent encore être reliés aux autres agences du groupe. Seul le trafic entre Lyon et Madagascar utilise le nouveau VPN IPsec.

Lors de la bascule des deux sites, nous avons réalisé les tâches suivantes :

Tableau 10 : Étapes de migration

N°	Titre de l'étape	Détail
1	Couper l'ancien pare-feu	Extinction complète
2	Câblage	Brancher les câbles réseau sur le nouveau serveur
3	Changement du nom	Pour que CFEngine installe les logiciels nécessaires au fonctionnement du pare-feu, il faut utiliser le mot-clé « fw » dans le nom de la machine
4	Mise à jour CFEngine	On force la mise à jour
5	Changement des règles OpenVPN	On applique les règles préparées dans l'étape 8 précédente
6	Vérification des accès locaux	Accès Internet sur le réseau des utilisateurs, en wifi, sur les machines virtuelles. Accès interne entre les réseaux précédents
7	Vérification des accès vers OpenVPN	Accès à l'agence de Paris – rue de Provence, accès aux autres sites
8	Vérification du VPN IPsec	Test d'accès d'une ressource située sur l'autre site, avec monitoring réseau

Les premiers tests se sont révélés concluants : le trafic entre les deux sites passait bien par le tunnel IPsec. De plus, OSPF ajoutait bien les routes disponibles sur le pare-feu distant. La latence entre les deux sites a été légèrement diminuée après l'activation du nouveau tunnel, ce qui prouve que la connexion se fait directement sans passer par le nœud central qui était l'agence de Paris – rue de Provence.

5.2.2 L'ajout de sites : un processus itératif

L'ajout de site dans le nouveau VPN n'est pas très différent des étapes citées précédemment. Le principal changement se trouve côté matériel : si l'agence possède un second équipement qui est utilisé pour le VPN IPsec, le paramétrage est simple et fiable et la préparation est réalisée en amont. Il est aisé de faire machine arrière en cas de problème lors de la migration finale en rebranchant l'autre serveur. Cependant, si nous devons mettre à jour l'équipement utilisé actuellement en production, il est souhaitable de réaliser une sauvegarde des données sensibles, à savoir les paramétrages réseau et les règles de routage actuelles, avant de lancer la mise à jour du système.

5.3 Résumé des problèmes techniques rencontrés

5.3.1 OVH un cas particulier

Pour l'agence d'OVH, nous avons été confrontés à un tout autre problème lié à la disponibilité de nos hébergements. OVH nous fournit un « Cloud Privé » fonctionnant sous VMware qui héberge environ 150 machines virtuelles. Ces machines sont *en production* et ne peuvent pas être débranchées

du réseau plus de quelques secondes. En revanche, le VPN qui relie cette infrastructure avec nos différents sites n'est pas critique, car il n'impacte pas les outils en production, il permet seulement d'administrer ces machines de manière sécurisée.

Nous avons préparé une seconde machine virtuelle « pare-feu » sur laquelle nous avons installé tous les outils habituels des pare-feu via CFEngine. Nous avons ensuite réalisé différents tests d'accès réseau pour valider le bon fonctionnement de cette nouvelle plateforme. Après avoir planifié l'intervention avec nos clients, nous avons transféré les adresses IP publiques de production vers la nouvelle machine. La coupure n'a duré que quelques secondes et nous n'avons pas rencontré de problèmes majeurs lors de cette phase de migration.

Nous avions décidé de relancer le système OpenVPN dans un premier temps pour réaliser cette migration pour conserver un périmètre équivalent. Quelques jours plus tard, nous avons activé les premiers tunnels IPsec entre OVH, Lyon et Madagascar avec succès.

5.3.2 Le site de Paris – rue de Provence : une criticité accrue

Le site situé rue de Provence est un cas particulier à plusieurs niveaux. Tout d'abord, car les services RH et comptabilité sont basés en partie sur ce site, mais aussi à Madagascar. Le VPN est utilisé par les collaborateurs Malgaches pour se connecter à leurs logiciels habituels. Ensuite, parce que c'est le cœur du réseau OpenVPN. Une coupure VPN empêcherait les développeurs d'accéder à des ressources distantes comme les bases de données par exemple. Enfin, plusieurs services principaux se trouvent sur ce site : annuaire LDAP, serveur DNS, outils de supervision, etc. Nous avons des copies de chacun de ces outils, sur plusieurs de nos sites, mais ils sont majoritairement dépendants de ceux installés rue de Provence.

Pour toutes les raisons évoquées précédemment, nous avons choisi de migrer ce site en dernier. Il marquera l'arrêt complet de l'ancien système. Cette migration n'est pas encore planifiée à l'heure actuelle, mais devrait intervenir avant la fin du premier semestre.

5.4 Transfert de compétences

5.4.1 Création de commandes utiles

Le nombre de tunnels GRE et IPsec augmentent à chaque migration d'agence. Pour ne pas être pollués par ces nouvelles connexions, nous avons mis en place quelques commandes simples qui filtrent et formatent les résultats qui sont affichés à l'écran. Ces commandes sont écrites dans le langage BASH et sont copiées sur tous les routeurs de l'entreprise à l'aide de CFEngine. Chacune d'entre elles renvoie uniquement les informations de la machine sur laquelle elle est exécutée.

5.4.1.1 Interfaces réseau

La liste des interfaces réseau s'est fortement alourdie avec GRE, mais aussi à cause de la segmentation de notre réseau en sous-parties logiques pour cloisonner les utilisateurs et les VM dans différents groupes dans le but d'augmenter notre sécurité face aux différentes menaces. Un routeur standard de Netapsys possède en général 6 interfaces réseau physiques, mais aussi plusieurs VLAN et enfin les tunnels GRE. Au total cela représente environ 20 interfaces réseau tous types confondus. La première commande a donc pour but d'afficher ces interfaces par groupe : elle se nomme « myip ».

```
root@fw01:/home/ssoubeyrand# myip --help
Usage: myip gre|notgre|eth|wan
```

Cette commande accepte un paramètre qui représente les données à afficher : *gre, notgre, eth* ou *wan*. Le code de cet outil est assez simple : nous commençons tout d'abord par lister les interfaces réseau actives. Puis nous allons parcourir cette liste et afficher les détails de l'interface si celle-ci correspond au critère du paramètre. Le résultat est directement visible dans la console comme nous pouvons le voir avec l'exemple suivant avec les liens Internet de notre agence de Lyon.

```
root@fw01:/home/ssoubeyrand# myip wan
Interfaces WAN :
6: eth4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:90:0b:44:d0:86 brd ff:ff:ff:ff:ff:ff
    inet 92.92.92.220/24 brd 92.92.92.255 scope global eth4
       valid_lft forever preferred_lft forever
23: ppp0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1456 state UNKNOWN group default qlen 3
    link/ppp
    inet 80.80.80.253 peer 193.193.193.3/32 scope global ppp0
       valid_lft forever preferred_lft forever
```

5.4.1.2 Surveiller le VPN IPsec

Avec le routage dynamique et le maillage complet entre les sites mis en place, notre système est assez performant pour que la coupure d'un tunnel IPsec n'impacte pas les accès des utilisateurs : une nouvelle route est calculée par OSPF au bout de quelques secondes de coupure et cet itinéraire *bis* et propagé sur tout le réseau.

Pour surveiller notre réseau IPsec, nous avons créé une commande qui compare ce qui est configuré avec ce qui fonctionne réellement. Cette commande « `check_vpn` » est écrite de la manière suivante :

```
#!/bin/sh
# Affichage de l'état des tunnels IPSEC en comparant les tunnels UP et le fichier ipsec.conf
NB_TUNNELS_INSTALLED=$(cat /etc/ipsec.conf | grep "conn" | awk '$2!="%default" {print $2}')
NB_TUNNELS_CREATED=$(ipsec statusall | grep "INSTALLED, TRANSPORT" | awk -F "{" '{gsub (" ", "", $1); print $1}')
for TUN_INS in $NB_TUNNELS_INSTALLED ; do
    Flag="0"
    for TUN_CRE in $NB_TUNNELS_CREATED ; do
        if [ "$TUN_INS" = "$TUN_CRE" ] ; then
            Flag="1"
        fi
    done

    if [ "$Flag" = "1" ] ; then
        echo "Tunnel $TUN_INS OK"
    else
        echo "Tunnel $TUN_INS non trouvé"
    fi
done
```

Tout d'abord, nous récupérons la liste des tunnels configurés via CFEngine (« `NB_TUNNELS_INSTALLED` ») puis nous récupérons les tunnels actifs (« `NB_TUNNELS_CREATED` »). Ces deux listes sont comparées et nous affichons le statut de chaque tunnel.

5.4.1.3 Surveiller OSPF

OSPF possède une commande identique à celle évoquée précédemment. Nous comparons la configuration d'OSPF aux informations récupérées par Quagga et nous les comparons. Voici le résultat de la commande sur le routeur de l'agence de Lyon.

```
root@fw01:/home/ssoubeyrand# check_ospf
Routeur 80.80.80.80 OK
Routeur 40.40.40.40 OK
```

5.4.1.4 Supervision

Les deux dernières commandes disposent d'une variante qui réalise les mêmes opérations, mais n'affiche qu'une seule ligne de résultat sous la forme « Tunnels IPsec : 5/5 » pour la première et « Routeurs OSPF : 5/5 » pour la deuxième. Les chiffres correspondent au nombre de tunnels IPsec/routeurs OSPF actifs en les comparant à ce qui est attendu. Ces commandes sont lancées toutes les 5 minutes par notre système de supervision. Si les deux nombres ne sont pas identiques, une alerte s'affiche en plus d'un mail envoyé aux membres de l'équipe infrastructure.

5.4.2 Documentation

Chez Netapsys, nous utilisons des outils collaboratifs pour rédiger les documentations. C'est donc dans la section « infra » de notre Wiki que j'ai ajouté les pages de documentation, dont un descriptif se trouve dans le *Tableau 11*.

Tableau 11 : liste des documents créés sur le Wiki

Titre	Contenu
VPN IPsec de Netapsys	Page d'introduction avec descriptif générique du VPN et du routage dynamique
Fonctionnement technique	Décris où trouver les fichiers de configuration dans CFEngine Décris les actions qui sont effectuées sur les routeurs lors des mises à jour de la configuration
Ajouter un sous-réseau	Procédure pour ajouter un nouveau sous-réseau dans une agence.
Ajouter un nouveau routeur	Liste des actions à effectuer pour ajouter un nouveau routeur dans le VPN Netapsys
Migrer un routeur vers le nouveau VPN	Liste les étapes que nous avons évoquées dans le paragraphe 5.2.2 sur l'ajout des sites.

Pour le moment, le sujet est encore très récent et mes collègues n'ont pas eu le temps d'effectuer beaucoup d'opérations sur le nouveau VPN. Dès que la mise en production sera terminée, je suis sûr qu'ils apporteront leurs commentaires et leurs remarques sur ces documents.

Chapitre 6

Bilan et perspectives

6.1 Perspectives du projet

6.1.1 Améliorations et évolutions

Tout au long du développement de ce nouvel outil, nous avons analysé le fonctionnement des communications réseau de l'entreprise. Dans cet objectif, nous avons étudié et analysé toutes les interactions possibles entre les différents sites afin de vérifier qu'ils soient pris en charge dans notre solution. Cette mission nous a permis d'identifier plusieurs points de faiblesse qui seront corrigés à l'avenir par l'équipe infrastructure.

En premier lieu, nous envisageons de doubler les pare-feu sur les sites les plus critiques, à savoir les sites de Lyon, Paris (rue de Provence et rue de Mogador) et sur notre Cloud Privé OVH. Ce second serveur sera préparé pour prendre le relai en cas de panne de la machine principale. Mes collègues ont déjà mis en place des solutions de haute disponibilité dans d'autres projets et apporteront leur expérience sur les outils à utiliser. Il nous faudra cependant veiller à ce que le système VPN IPsec et OSPF s'intègrent dans ces solutions.

Pour IPsec, StrongSwan dispose d'un module « *High Availability* » (haute disponibilité) qui peut fonctionner en mode actif-passif[17] ou en mode actif-actif[18]. Le premier mode correspondrait mieux à notre besoin, car certaines de nos connexions Internet sont difficilement exploitables par deux serveurs en même temps (mode actif-actif).

Pour OSPF, Quagga ne gère à priori aucun de ces deux modes. Toutefois, nous pouvons surement trouver une solution qui consisterait à faire fonctionner un serveur OSPF sur chaque machine. La machine « en attente » ne sera pas reliée à Internet, et donc l'algorithme de recherche du plus court chemin d'OSPF ne donnerait aucune route à partir de ce routeur. Lors de la bascule vers ce routeur secondaire, les connexions Internet seraient restaurées ainsi que les tunnels IPsec sur le serveur en attente. Les tunnels GRE étant basés sur IPsec, ils seraient actifs à leur tour sans intervention humaine. Enfin, le routeur OSPF « en veille » sera promu dans le nouveau calcul SPF et les connexions réseaux routées automatiquement via l'équipement de secours.

[17] https://www.strongswan.org/testing/testresults/ha/active-passive/index.html

[18] https://www.strongswan.org/testing/testresults/ha/both-active/index.html

Bien sûr, plusieurs étapes de tests seront nécessaires pour valider ce type d'architecture.

Pour améliorer la disponibilité de nos différents sites, il sera surement nécessaire à terme d'ajouter une seconde connexion capable d'absorber le trafic nécessaire suite à une coupure du lien principal. Avec nos tests, nous savons que notre système est déjà compatible avec cette solution (*4.3.1.2 - Plateforme de test avancée*). Avec l'aide de CFEngine, nous pouvons aisément injecter les paramètres de connexions Internet de secours en leur attribuant un faible cout dans le calcul du plus court chemin SPF.

La dernière faiblesse que nous avons identifiée se trouve en amont du projet évoqué ici : il s'agit du serveur de gestion des versions (SVN) qui contient le code de CFEngine. Ce serveur, qui se trouve actuellement sur le site de Provence, est un point de défaillance unique. Une copie disponible sur un autre site avec une bascule manuelle (ou automatisée si possible) me parait être une solution pérenne pour éviter les problèmes en cas de dysfonctionnement sur le site de Paris - rue de Provence.

6.1.2 La première brique d'un « *Software-defined networking* »

Mon responsable m'a fait remarquer récemment que ce projet pouvait s'apparenter à du *Software-Defined Networking* (SDN ou *Réseau Défini par le logiciel*). Après quelques recherches, je peux confirmer cette analyse. En effet, le SDN consiste à séparer les couches qui composent les équipements réseau afin de centraliser la couche logicielle. Le SDN est un concept-clé pour faire le pont entre la gestion dynamique des ressources réseau d'un côté et la demande en connectivité et en qualité de service des applications de l'autre côté.

La *Figure 24*, issu du livre de [Stallings, 2015], illustre le fonctionnement de ce nouveau réseau globalisé. Dans cette nouvelle approche, nous avons un seul logiciel pour contrôler tous les équipements réseau, peu importe leur marque et leur modèle.

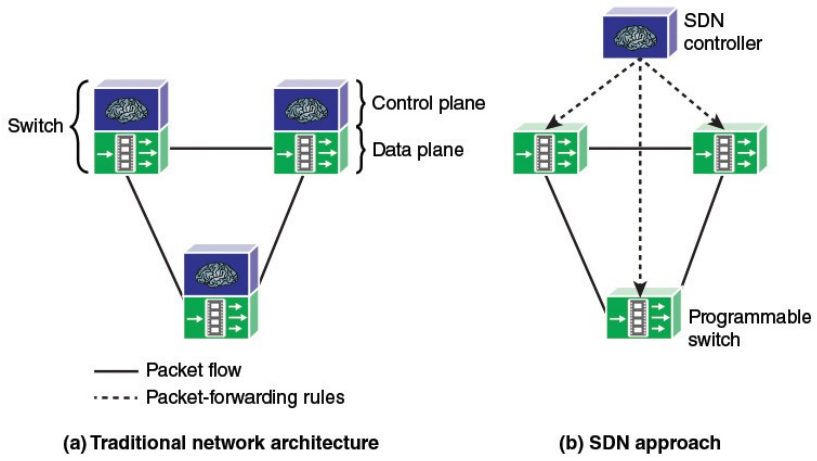


Figure 24 : Architecture SDN

Notre nouveau VPN, avec routage dynamique et piloté par CFEngine s'en approche fortement. Nous avons réalisé ce découpage entre matériel et logiciel à l'aide d'outils Open Source et de matériels hétérogènes fonctionnant sous Linux.

La prochaine étape sera surement de s'approprier les nouveaux concepts SDN normalisés et de rendre notre système compatible avec les outils existants. Nous pouvons aussi envisager d'autres évolutions comme une extension de ce VPN vers les postes utilisateurs nomades ou encore l'ajout

d'une couche pour améliorer la qualité de service en priorisant certaines données par rapport aux autres (QoS – *Quality of Service*).

6.2 État du projet

Le projet « évolution du système VPN », objet de ce mémoire, a commencé il y a presque un an. On constate un dépassement de plusieurs mois par rapport aux prévisions du calendrier initial : la fin du projet était attendue pour décembre 2015. Ce projet n'étant pas critique vu qu'il s'agit d'une évolution de notre architecture interne, le retard n'est pas préjudiciable à l'entreprise.

Nous sommes actuellement en phase de déploiement des sites vers le nouveau VPN. Il est encore un peu tôt pour évoquer la fin de projet. Cependant, d'après les premières migrations que nous avons lancées, les premiers retours sont positifs. Nous n'avons pas rencontré de problèmes majeurs, seulement des ajustements. Quelques mois supplémentaires d'utilisation en production pourront confirmer cela.

Sans attendre la fin du déploiement, nous pouvons affirmer que le projet est bénéfique pour l'entreprise Netapsys à plusieurs niveaux. Tout d'abord, le travail effectué a eu pour conséquence de lancer un grand nettoyage des pare-feu utilisés. La migration vers le nouveau système nous a obligés à mettre au propre des logiciels qui étaient montés un peu rapidement. Je pense par exemple au serveur utilisé à Madagascar, qui possédait une configuration quasi exclusivement manuelle. À présent, celui-ci est paramétré via CFEngine et s'inscrit entièrement dans notre infrastructure. Les systèmes d'exploitation utilisés par ces machines sont aussi tous alignés. Cette nouvelle base nous ouvre les portes de futures évolutions.

Du point de vue de règles d'accès réseau, nos différents tests ont mis en lumière la complexité des échanges entre les sites de l'entreprise. En effet, la société est en train de changer de dimension depuis les précédentes acquisitions. La segmentation du réseau en sous-groupe est une évolution nécessaire pour une entreprise comme la nôtre, car la sécurité des données et des utilisateurs doit être prise en compte. Ce découpage engendre une grande quantité de paramétrages supplémentaires qui sont regroupés, avec notre solution, dans seulement quelques fichiers.

Enfin, depuis fin 2015, l'entreprise Netapsys a est rachetée par le groupe SODIFRANCE. Le nouveau groupe ainsi formé représente 1250 collaborateurs. Notre président actuel, M. Yoan Hébert, est devenu le directeur général du nouveau groupe. Les deux entreprises sont encore au stade du rapprochement. La première phase, qui a déjà débuté, vise à créer des synergies au sein des équipes commerciales.

Nous avons reçu une présentation du système d'information de SODIFRANCE il y a quelques jours. Leur réseau actuel fonctionne à l'aide de liens MPLS et le site de Nantes est la passerelle d'accès Internet pour tous les autres sites. Suite à cette présentation et à l'achèvement de mon mémoire, mon responsable, M. Simone, souhaiterait que nous présentions notre nouveau système VPN au directeur des systèmes d'informations de SODIFRANCE.

6.3 Bilan personnel

Je retire de ce projet « *évolution du système VPN de l'entreprise Netapsys* » un enrichissement personnel et professionnel. Le fait de mener ce projet m'a permis de développer mes compétences professionnelles, à la fois en termes d'architecture réseau, d'approfondissement du langage Linux et

de CFEngine. Maintenant que ma mission est presque terminée, je peux prendre du recul, grâce en particulier, à la rédaction de ce mémoire.

Pendant le déroulement du projet, j'ai dû faire face à quelques difficultés ou retards. Les difficultés techniques ont été résolues grâce à l'expertise que j'ai acquise au fur et à mesure de la prise en main des produits employés.

Enfin, ce sujet de mémoire a fait appel à des technologies variées pour lesquels l'entreprise ne disposait pas de référent/de document technique. La contrepartie de cette « libre » expérimentation a été ressentie successivement comme une grande liberté, une petite solitude, mais au final, je réalise que j'ai gagné en autonomie et j'ai compris que la tâche d'un ingénieur ne nécessitait pas seulement une expertise technique. Il doit aussi savoir gérer l'organisation complète d'un projet, ainsi que les répercussions des changements apportés par la mise en place de nouvelles technologies. Il me faudra encore améliorer mes capacités d'anticipation et de synthèse.

L'expérience acquise durant ces 10 mois me permet de dire que j'aurais pu, pour éviter certains écueils, utiliser plus efficacement le temps passé à l'étude théorique, pour m'efforcer de mieux anticiper les problèmes techniques, liés au déploiement, que j'ai rencontrés. Ce que je ferai à l'avenir. Cette mission termine le cycle de formation de 5 années passées au CNAM, et m'a permis de mener à bien ma première mission d'ingénieur.

Conclusion

Dans le cadre de ce projet visant à faire évoluer le VPN de l'entreprise Netapsys, nous avons établi un état de l'art des technologies employées pour améliorer le *Réseau Privé Virtuel* existant et ajouter le routage dynamique entre les sites de l'entreprise. Cette démarche nous a permis de sélectionner les protocoles IPsec et OSPF comme étant les plus adaptés à notre besoin. Après une recherche des meilleurs logiciels couvrant notre besoin, nous avons entamé la phase technique avec la mise en place de plateforme de tests. Tout d'abord, en ayant recours à des machines virtuelles, puis dans une deuxième phase en déployant une première architecture multisite. Ces étapes nous ont permis de valider le paramétrage des différentes applications et d'appréhender l'industrialisation de notre solution à l'aide de CFEngine. Lors de la phase de déploiement du nouveau système, nous avons activé le 1er VPN entre les sites de Lyon et de Madagascar, ce que nous avons ensuite répété sur les autres sites de l'entreprise.

Les principales difficultés rencontrées, d'ordre technique, ont été résolues durant les phases de test. À ce jour, le déploiement n'est pas terminé, nous pensons toutefois, avoir identifié et minimisé les principales sources de problèmes. Le déploiement effectif, qui sera terminé à la fin du premier semestre, permettra de réaliser un bilan plus complet du dispositif.

Enfin, la portée de ce nouvel outil dépendra de la solution retenue dans le nouveau groupe formé par les entreprises Netapsys et SODIFRANCE.

Bibliographies

Agence nationale de la sécurité des systèmes d'information (ANSSI), 2015. *Recommandations de sécurité relatives à IPsec pour la protection des flux réseau.* [Online]
Available at: http://www.ssi.gouv.fr/uploads/2012/09/NT_IPsec.pdf
[Accessed 08 04 2016].

Cameron, R. & Woodberg, B., 2013. *Juniper SRX Series.* Sebastopol, Californie: O'Relly Media.

Computer Center, Peking University, 2010. *IPsec and OpenVPN worked-out examples.* [Online]
Available at: http://ipseclab.eit.lth.se/tiki-index.php?page=6.%20OpenVPN
[Accessed 31 03 2016].

Datatracker IETF, 2016. *Cisco Enhanced Interior Gateway Routing Protocol - draft-savage-eigrp-05.* [Online]
Available at: https://datatracker.ietf.org/doc/draft-savage-eigrp/
[Accessed 20 03 2016].

Frankel, S. et al., 2005. *Guide to IPsec VPNs.* Special Publication 800-77 ed. Gaithersburg, Maryland: National Institute of Standards and Technology.

Latu, P., 2015. *Inetdoc.net, Interconnexion réseau et logiciel libre.* [Online]
Available at: http://www.inetdoc.net/guides/
[Accessed 28 04 2016].

Mogollon , M., 2008. *Cryptography and Security Services: Mechanisms and applications.* Hershey, Pennsylvanie: IGI Global.

Palomares Velasquez, D., 2013. *Etude de Mécanismes Assurant la Continuité de Service de Protocoles IKEv2 et IPsec.* [Online]
Available at: https://tel.archives-ouvertes.fr/tel-00939092
[Accessed 12 03 2016].

Sequeira, A., 2013. *Interconnecting Cisco Network Devices, Part 1 (ICND1) Foundation Learning Guide.* 4th ed. Indianapolis: Cisco Press.

Shamim, F., Aziz, Z., Liu, J. & Martey, A., 2002. *Troubleshooting IP Routing Protocols.* Indianapolis, Indiana: Cisco Press.

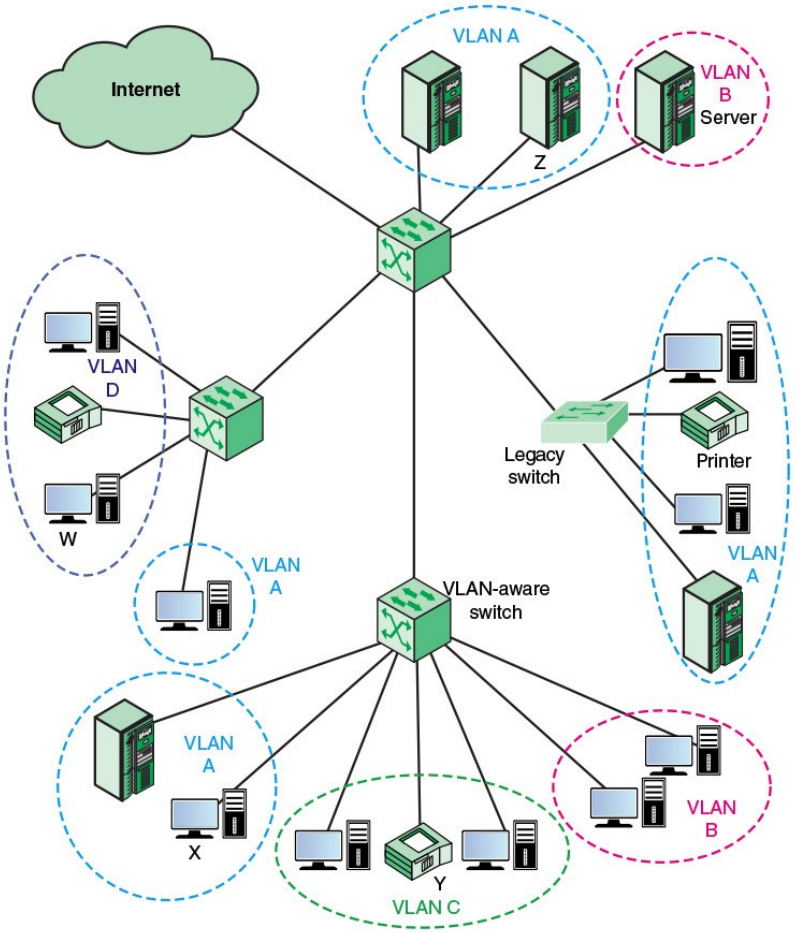
Stallings, W., 2015. *Foundations of Modern Networking.* Indianapolis, Indiana: Pearson.

Tadimety, P. R., 2015. *OSPF: A Network Routing Protocol.* New York: Apress.

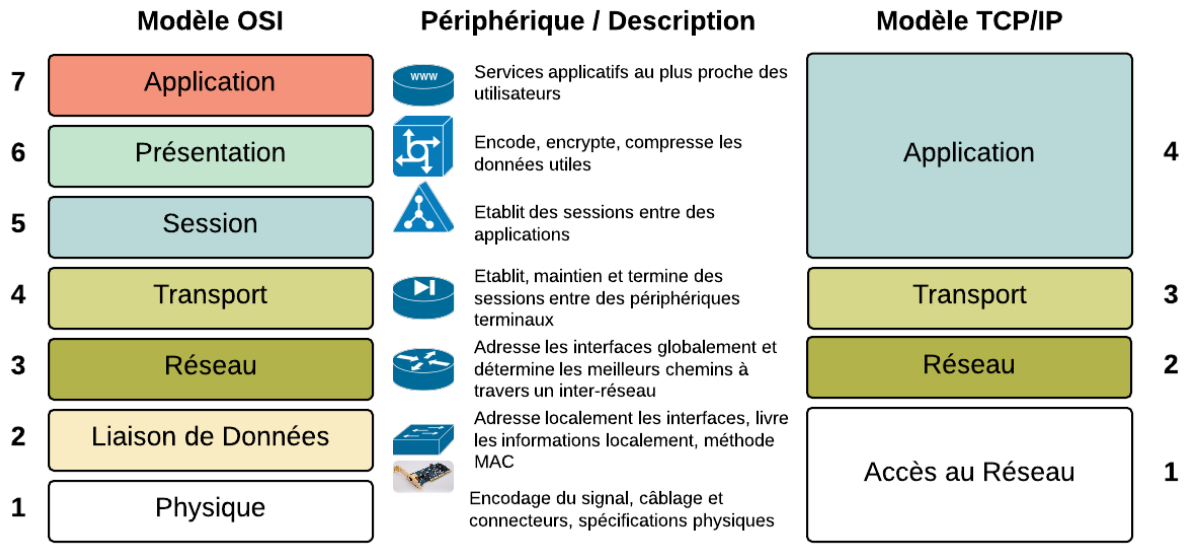
Table des annexes

Annexe I : Exemple de découpage d'un réseau Informatique à l'aide de VLAN 64
Annexe II : Modèle OSI 65
Annexe III : Protocoles utilisés par IPsec [Mogollon , 2008] 65
Annexe IV : Trame IP encapsulé avec IPsec/ESP 66
Annexe V : Structure complète du message OSPF/Hello 66
Annexe VI: Entête d'un paquet GRE 67
Annexe VII : Fonction GenerateAllTunnelsGre 68
Annexe VIII : Fonction GenerateMyTunnelsGRE 69
Annexe IX : Script de peuplement des groupes de routage avec ipset 70

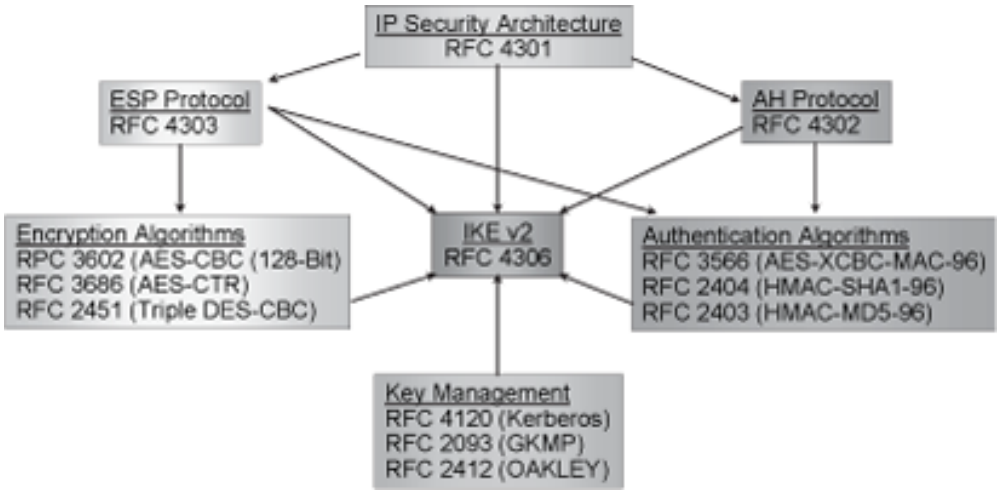
Annexe I : Exemple de découpage d'un réseau Informatique à l'aide de VLAN



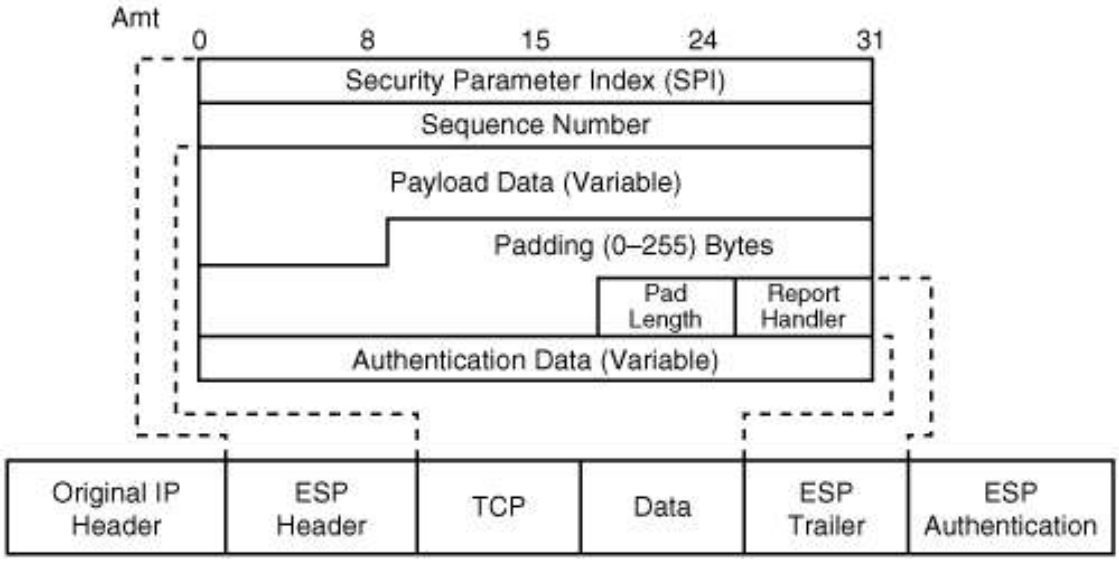
Annexe II : Modèle OSI



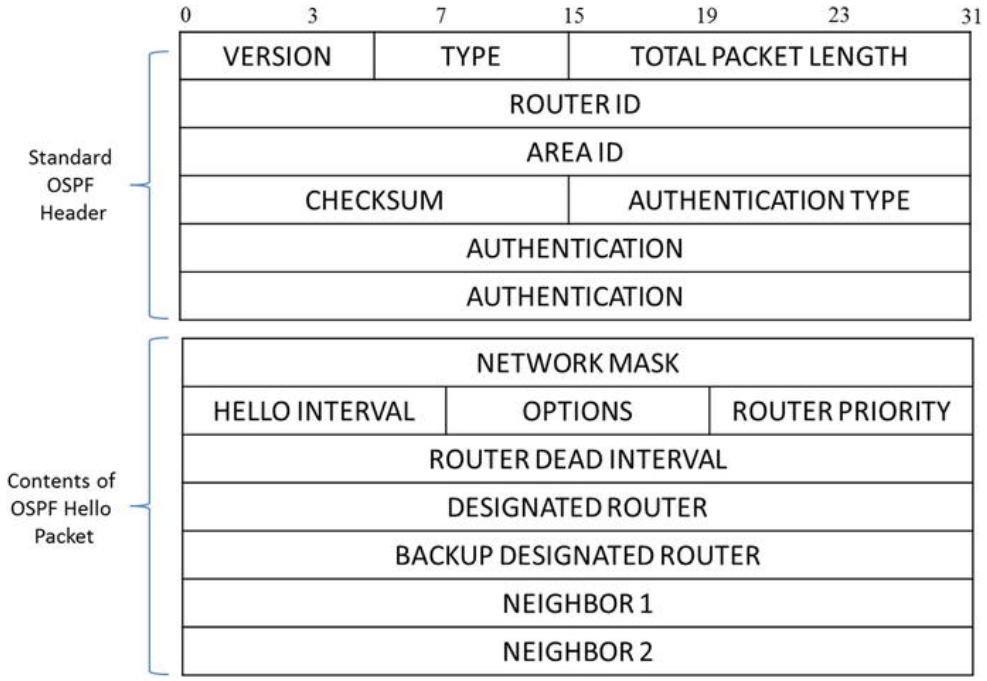
Annexe III : Protocoles utilisés par IPsec [Mogollon , 2008]



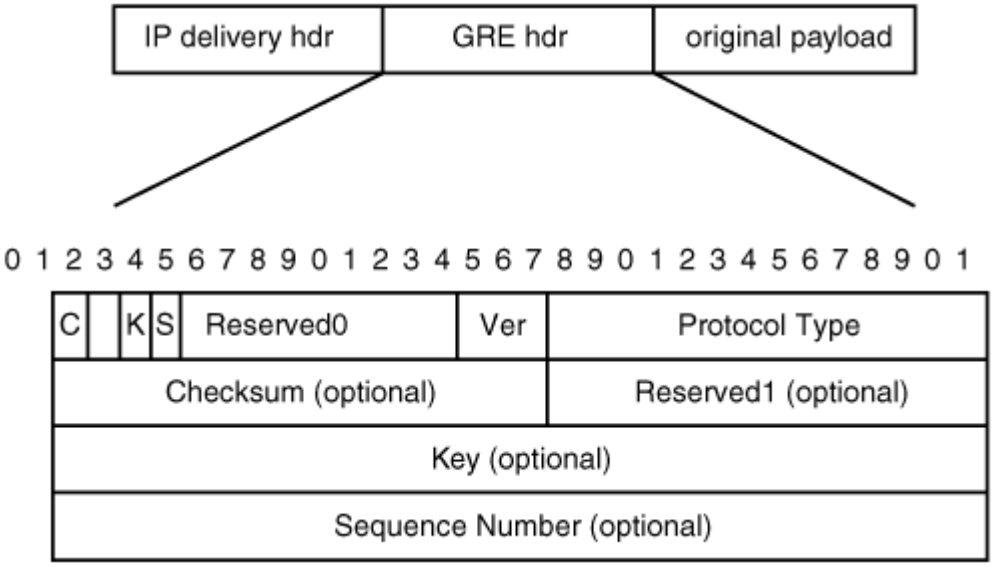
Annexe IV : Trame IP encapsulé avec IPsec/ESP



Annexe V : Structure complète du message OSPF/Hello



Annexe VI: Entête d'un paquet GRE



Annexe VII : Fonction GenerateAllTunnelsGre

```
# Fonction qui génère toutes les combinaisons possibles en éliminant
# les tunnels sur les mêmes FW (ra01-ra02 par exemple)
function GenerateAllTunnelsGRE () {

    #Initialisation des variables
    SortedWan=$(GetSortedWanName)
    TunnelsToDelete=($(cat $jsonfile | jq ".tunnelstodelete[].tunnel" -r))
    i=1

    # on part du cas parfait où tout le monde se connecte à tout le monde
    for Left in $SortedWan ; do
        for Right in $SortedWan ; do
            if [[ "$Left" < "$Right" ]] ; then
               # récupération du firewall correspondant à chaque wan
               RouterLeft=$(GetRouterNameByWanName $Left)
               RouterRight=$(GetRouterNameByWanName $Right)
               # Si c'est un firewall différent, je continue
               if [[ "$RouterLeft" != "$RouterRight" ]] ; then
                   # renvoi d'une ligne contenant toutes les infos nécessaires pour le tunnel GRE
                   echo "$Left-$Right-$(GetGreIP $i Subnet)-$(GetGreIP $i Local)-$(GetGreIP $i Remote)"
                   ((i++))
                fi
            fi
        done
    done
}
```

Annexe VIII : Fonction GenerateMyTunnelsGRE

```
# Fonction qui récupère toutes les tunnels et ne conserve que ceux qui correspondent à mon routeur
function GenerateMyTunnelsGRE () {
    # Tous les tunnels possibles
    for tunnel in $AllTunnels ; do
        #pour chaque connexion WAN locale
        for wan in $MyWans ; do
            #si le tunnel contient mon WAN
             if [[ $tunnel == *"$wan"* ]] ; then

                # je découpe le résultat
                IFS="-" read sideA sideB subnet sideAip sideBip <<< $tunnel

                # Est-ce qu'on est coté serveurA ou serveurB ? (C'est pour les adresses IP à renvoyer)
                if [[ "$sideA" == "$wan" ]] ; then
                    #oui: ServeurA / je renvoie à l'identique
                    echo $tunnel
                else
                    # non, on est coté serveurB : j'inverse les infos du tunnel
                    echo "$sideB-$sideA-$subnet-$sideBip-$sideAip"
                fi
            fi
        done
    done
}
```

Annexe IX : Script de peuplement des groupes de routage avec ipset

```
#/bin/bash
#-------------------------------------------------------------------------------------------------------------
# Génération des règles iptables en utilisant :
# - Fichier json utilisé par le VPN IPSEC
# - Fichier json des groupes personnalisés
#-------------------------------------------------------------------------------------------------------------

# Variables
ipsec_servers_jsonfile="/usr/local/sbin/ipsec_servers.json"
iptables_groups_jsonfile="/usr/local/sbin/iptables_groups.json"

# affichage des groupes dans la sortie, c'est plus simple à lire
echo "#/bin/bash
#------------------
# Généré par le script iptables_create_rules.sh / ne pas modifier à la main
#------------------
# ipset créés
"


#-------------------------------------------------------------------------------------------------------------
# Récupération des groupes dans le fichier JSON du VPN IPSEC

GroupesIPSEC=$(cat $ipsec_servers_jsonfile | jq '.ipsecsrv[].lan[].group, .ipsecsrv[].lanindirect[].group' -r | sort -u)


# pour chaque groupe,
for Groupe in $GroupesIPSEC ; do
    # récupération des subnets associés à ce groupe
    temp_chain=($(cat $ipsec_servers_jsonfile | jq '(.ipsecsrv[].lan[] | select(.group==$mygroup) | .subnet) , (.ipsecsrv[].lanindirect[] |
select(.group==$mygroup) | .subnet)' --arg mygroup $Groupe -r ))
    # Création du groupe ipset
    echo "ipset create $Groupe hash:net -exist"
    # je vide les données (s'il existait déjà)
    echo "ipset flush $Groupe"
```

```
    # Boucle sur les éléments résulats pour remplir l'ipset
    for ((i=0; i<${#temp_chain[@]}; ++i)) ; do
        # ajouter l'élément dans le tableau
        echo "ipset add $Groupe ${temp_chain[$i]}"
    done
    echo ""
done


#-----------------------------------------------------------------------------------------------------
# Récupération des groupes personnalisés dans le fichier JSON dédié


GroupesPERSO=$(cat $iptables_groups_jsonfile | jq '.groups[].name' -r | sort -u)


# pour chaque groupe,
for Groupe in $GroupesPERSO ; do

    # récupération des subnets associés à ce groupe
    temp_chain=($(cat $iptables_groups_jsonfile | jq '.groups[] | select(.name==$mygroup) | .subnets[].subnet ' --arg mygroup $Groupe -r ))
    temp_comment=($(cat $iptables_groups_jsonfile | jq '.groups[] | select(.name==$mygroup) | .subnets[].description ' --arg mygroup $Groupe -r ))

    # Création du groupe ipset
    echo "ipset create $Groupe hash:net -exist comment"

    # je vide les données (s'il existait déjà)
    echo "ipset flush $Groupe"

    # Boucle sur les éléments résulats pour pour remplir l'ipset

    for ((i=0; i<${#temp_chain[@]}; ++i)) ; do
        # ajouter l'élément dans le tableau
        echo "ipset add $Groupe ${temp_chain[$i]} comment \"${temp_comment[$i]}\""
    done
    echo " "
done
```

Table des figures

Figure 1 : Évolution du nombre de collaborateurs de Netapsys 8
Figure 2 : Croissance du chiffre d'affaires de Netapsys et évènements clés 9
Figure 3 : Architecture de CFEngine chez Netapsys 11
Figure 4 : Entreprise utilisant des connexions louées 20
Figure 5 : Entreprise utilisant un VPN via Internet 20
Figure 6 : Fonctionnement d'OpenVPN 21
Figure 7 : Client-Serveur avec OpenVPN 22
Figure 8 : Ancien VPN de Netapsys 23
Figure 9 : Mode IPsec "transport" 24
Figure 10 : Mode IPSEC "tunnel" 24
Figure 11 : Trame IP en mode IPsec "transport" 25
Figure 12 : Trame IP en mode IPSEC "tunnel" 25
Figure 13 : IKE : les échanges de données lors de l'établissement de la connexion 26
Figure 14 : Types de protocoles de routage dynamique [Sequeira, 2013] 27
Figure 15 : Unicast, Broadcast et Multicast 29
Figure 16 : Algorithme de Dijkstra 29
Figure 17: Zones OSPF [Tadimety, 2015] 32
Figure 18 : Encapsulation des données via GRE puis via IPsec 33
Figure 19 : Paquet IPsec + GRE 34
Figure 20 : Premier environnement de test avec des machines virtuelles 40
Figure 21 : Représentation logique des tunnels IPsec du premier test 41
Figure 22 : Test avec des liens Internet doublés 42
Figure 23 : Représentation logique des tunnels IPsec du second test 43
Figure 24 : Architecture SDN 58

Liste des tableaux

Tableau 1 : Les parties prenantes du projet 15
Tableau 2 : Planning du projet initial 18
Tableau 3 : Diagramme de Gantt du projet 19
Tableau 4 : Comparaison des protocoles de routage dynamique 28
Tableau 5 : Types de paquets OSPF 31
Tableau 6 : Détail du premier sous-réseau pour les tunnels GRE 46
Tableau 7 : Inventaire des pare-feu de Netapsys 50
Tableau 8 : Pare-feu uniformisés 51
Tableau 9 : Préparation à la migration 52
Tableau 10 : Étapes de migration 53
Tableau 11 : liste des documents créés sur le Wiki 56

EVOLUTION DU SYSTEME VPN DE L'ENTREPRISE NETAPSYS

MEMOIRE PRESENTE EN VUE D'OBTENIR LE

DIPLÔME D'INGÉNIEUR CNAM

SPECIALITE : INFORMATIQUE

LYON, 2016

RÉSUME

L'entreprise Netapsys souhaite faire évoluer son système VPN qui relie ses différents sites. Notre solution se base sur les protocoles IPsec pour garantir la sécurité des échanges avec une architecture en maille pleine. OSPF apporte le routage dynamique entre les équipements du réseau dans le but d'automatiser le partage des informations de routage. Les tunnels GRE, qui s'insèrent entre IPsec et OSPF fournissent la couche multicast nécessaire aux échanges OSPF. Ces services sont pilotés avec le logiciel de gestion des configuration CFEngine. Enfin, le déploiement est planifié après plusieurs tests de la solution sur différentes plateformes.

Mots-clés : réseau informatique, VPN, IPsec, protocole de routage dynamique, OSPF, Linux, Open Source, CFEngine, pare-feu, BASH

SUMMARY

Netapsys company wants to evolve its VPN system which connects its various sites. Our solution is based on the IPsec protocols to ensure the security of exchanges with a full meshed architecture. OSPF provides dynamic routing between network equipment to automate routing information sharing. GRE tunnels, which fit between OSPF and IPsec provide the necessary OSPF multicast layer. These services are driven with the CFEngine configuration management software. Finally, the deployment is planned after several tests of the solution on different environments.

Keywords : networking, VPN, IPsec, dynamic routing protocol, OSPF, Linux, Open Source, CFEngine, firewall, BASH