Conservatoire National des Arts et Métiers
Centre Régional Associé de Limoges

SPÉCIALITÉ : Ingénierie et Intégration Informatique
OPTION : Systèmes d'Information

MÉMOIRE
présenté en vue d'obtenir
le DIPLÔME d'INGÉNIEUR CNAM

Par

Jean-Philippe Gaulier
jpgaulier@point-libre.org

Mise en place d'une solution de ressources réparties et sécurisées composées de logiciels libres au sein d'une info-structure

Soutenu le
22 novembre 2007

Jury composé de
Présidente : Madame M.-C. Costa
Membres : Monsieur J.-J Charrière
Monsieur P. Jeulin
Monsieur D. Skrzypezyk
Monsieur A. Mascret

Table des matières

Liste des tableaux

Table des figures

Remerciements

La déontologie impose le remerciement de ses pairs et encadrants lors de la remise d'un rapport. Dire merci est aujourd'hui presque quelque chose d'anodin, voire dépassé. Je souhaite, pour ma part, faire preuve de gratitude envers toutes les personnes qui ont permis l'aboutissement de ce mémoire, à travers les mois de réalisation, de maquettage, d'écriture, de correction.

Je tiens à remercier particulièrement Alix Mascret pour l'implication qu'il a pu avoir tout au long de ce projet, son suivi, ses conseils, ses directions et éclaircissements. Après plus de six années de travail commun et quotidien, c'est toujours un réel plaisir d'être à ses côtés et de pouvoir travailler ensemble à l'aboutissement de nos rêves.

Je m'incline particulièrement devant Vincent Gallice, Marie-Françoise Gaulier et Éliane Lajoie qui semaine après semaine n'ont eu de cesse de me harceler pour que j'amène à terme cette étude. Votre acharnement aura permis la réalisation de ce travail.

Un document ne peut être remis sans lecture et relecture. Ainsi, un grand merci envers mes relecteurs, pour leurs remarques et conseils avisés : Jean-Pierre lainé, Bernadette Gaulier, Marc Falzon, Marc Sert, Éric De La Musse, Joelle Leconte, Stephan Nicolas, Guilhem Urroz, Nicolas Ledez et Vincent Batoufflet. Un merci particulier à Stéphane Dodeller pour sa constance dans les relectures.

Je remercie le jury pour l'attention qu'il m'apporte ainsi que Philippe Jeulin pour son tutorat.

Pour finir, je remercie mon épouse, Marjorie, ainsi que celui en qui se place ma foi, pour leur écoute quotidienne, leur attention et leur soutien. Sans eux, je ne serais rien.

Introduction

L'Internet aborde aujourd'hui un tournant, avec un grand *buzz* autour du Web2.0, notion économique qui tente de faire rêver les investisseurs comme lors de la ruée vers l'or durant la conquête des États-Unis d'Amérique. Cette appropriation du média par le grand public permet l'émergence d'usages jusqu'alors confidentiels. En effet, c'est bien sous l'impact du grand public que le peer-to-peer a explosé, que les carnets personnels (lire *blogs*) foisonnent. Les technologies ne sont pas nouvelles, mais leur appropriation, si. Devant cette popularisation, il est de bon aloi de se remémorer qu'Internet n'est pas un lieu magique, né spontanément d'un désir collectif, mais bien la résultante d'une boulimie de travail de quelques hommes et de femmes, de quelques organismes, privés et publics, qui se sont acharnés durant les trente dernières années à établir des concepts, créer des standards, des protocoles de communication et fournir du logiciel.

Dans ce paysage, l'École Ouverte Francophone (Éof) s'évertue à inventer un nouveau schéma de formation, mais se veut également une émulsion de la communauté du libre francophone. C'est au sein de cet organisme que se dessine ce mémoire.

Il m'a été confié l'étude de la mise en place d'une solution de ressources réparties et sécurisées à base de logiciels libres au sein d'une info-structure.

Les difficultés de cette étude ne se posent pas uniquement au niveau technique, bien qu'il soit toujours possible à court ou long terme, de maîtriser une technologie. Il est également complexe d'en définir le but, l'usage. Nous nous attarderons sur ce sujet, mais il convient pour cela de dessiner les courbes de l'Éof. En tant que centre de formation, l'école souhaite, entre autre, contribuer à l'avancement de la recherche scientifique, dans les domaines de la pédagogie [1] ou de la sécurité informatique. Dans le premier domaine, une collaboration forte a été mise en place avec le département Innovation Pédagogique de l'École Nationale Supérieure des Télécommunications de Paris, de laquelle a découlé une plate-forme de formation, Amarante. Dans le second domaine, l'école offre un module de sensibilisation et d'ouverture à la réflexion dans le domaine de la sécurité. Sur le long terme, elle souhaite pouvoir participer plus activement par l'intermédiaire de publications, basées sur des recherches et des expériences. De ce fait, la maquette présentée dans ce mémoire permettra à l'école d'établir les bases du travail qu'elle désire réaliser.

De nombreuses technologies vont être abordées, en commençant par les plus anciennes, comme le *dns* ou le *web*. En effet, il semble obligatoire de connaître le fonctionnement des protocoles basiques pour pouvoir analyser les usages présents et futurs. Ce sont ces mêmes usages qui posent problèmes en terme de sécurité, puisqu'à l'heure où les prémices d'un Web 3 apparaissent, il est de plus en plus fait fi des obligations de sécurité en faveur d'un développement et d'un déploiement *ultra* rapide. On peut constater la tendance avec le protocole IPv6 qui n'est actuellement proposé par aucun des grands acteurs du marché français de connexion à Internet, alors que les pays asiatiques et du moyen orient

prennent une avance considérable, en déployant d'ores et déjà leurs solutions. Le *pourquoi* s'impose avec une réponse simple : nous n'en avons pas l'usage.

Ce mémoire débute ainsi un travail se déroulant en trois parties. L'année d'étude et le maquettage concernent la première partie, c'est également le contenu de ce document. Dans sa seconde partie, le programme prévoit l'installation de l'architecture matérielle et logicielle, ainsi que le déploiement des applications de production et la formation des personnels. Ces applications, également appelés *middleware*, devront être traitées dans une autre étude. Dans la dernière partie, le process se focalisera sur le déploiement des aspects centrés sur la pédagogie de la plate-forme formation des personnels.

Une invitation au voyage, donc, à travers les technologies, leurs fondements, leurs usages et la sécurité comme passeport de transition.

Chapitre 1

Contexte du mémoire

I Présentation de l'École Ouverte Francophone

Après trois années d'étude et de recherche sur l'enseignement à distance, la recherche de financements, la mise en place d'échanges et de partenariats, l'École Ouverte Francophone voit le jour officiellement le 21 septembre 2005 à Limoges, lors de son inauguration à la technopole Ester. L'école affiche clairement ses objectifs : travailler par le libre, pour le libre et avec le libre. Pour cela, elle recrute une quinzaine d'enseignants qui produiront des supports et assureront le suivi des cours et des élèves. Ces enseignants participent à différents projets de la communauté du libre et sont issus de celle-ci.

La première session de certification démarre en janvier 2006, avec onze inscrits. Le cursus principal comprend l'installation d'un système, l'étude de langages de publication, de script (shell, python, PERL…), la méthodologie de conception, les gestionnaires de contenu, les services serveurs, le réseau et la sécurité. Par période de six mois, les élèves étudient à distance en vue d'obtenir la certification professionnelle délivrée par l'école.

Depuis le démarrage, de nouveaux cursus ont vu le jour :

- l'initiation *Débuter Sous Linux* (DSL) propose gratuitement pendant trois semaines un accompagnement à la découverte du poste de travail de l'utilisateur final GNU/Linux ;
- une autre formation est également destinée aux utilisateurs finaux, sur le sujet de la suite bureautique OpenOffice.org, présentée comme alternative aux suites privatives disponibles sur le marché ;
- la téléphonie sur IP, domaine jeune mais très riche, est abordée à travers la formation AS ; on y découvre les bases de la téléphonie, mais également la configuration d'un central téléphonique logiciel, Asterisk ;

En 18 mois, ce sont 120 personnes qui ont pu être formées grâce à l'école, en formation à distance.

II Les logiciels libres

Le logiciel libre est né en 1984, sous l'impulsion de Richard Matthew Stallman, fondateur du mouvement *GNU*. L'idée initiale est que tout logiciel devrait respecter 4 règles fondamentales, afin de protéger les *droits* de l'utilisateur :

- la liberté d'exécution, soit le droit de n'avoir aucune restriction dans les options utilisables, le temps ou encore le lieu (maison, bureau…) ;

- la liberté de consulter le code source, soit la possibilité d'étudier les entrailles d'un logiciel, à la fois les algorithmes, mais également l'implantation ;
- la liberté de modifier le code source, afin de pouvoir corriger, ajouter, modifier ou supprimer des bouts de code ;
- la liberté de redistribuer le logiciel, sans restriction.

Ces libertés sont énoncées dans la licence *GPL* (GNU Public Licence).

L'ensemble de ces quatre libertés définit ce qu'est un logiciel libre. La gratuité ne fait pas partie des critères de liberté, mais la quasi-totalité des logiciels répondant à ces critères sont disponibles sans aucune nécessité de rémunération. De nouveaux schémas économiques apparaissent dans cette communauté d'utilisateurs, appelée communauté du logiciel libre. Ceux-ci se basent essentiellement sur le paiement au service et non contre une licence d'utilisation.

Les logiciels dont le code source n'est pas accessible sont appelés *logiciels propriétaires*, en lien avec la propriété intellectuelle de leurs auteurs. Un logiciel dont l'accès au code source est disponible mais qui ne posséderait aucune des autres libertés peut être appelé *open source*. Dans tous les cas, un logiciel qui ne répond pas aux libertés fondamentales évoquées ci-dessus est appelé *logiciel privatif* car il prive les utilisateurs de leurs droits fondamentaux.

De nombreux utilisateurs se regroupent à travers la terre, sous la forme d'associations nommées *Linux User Group* (LUG). Les sociétés de services du domaine sont identifiées par le sigle SSLL (Société de Service en Logiciel Libre) et sont reconnues en tant que telles dès lors que des remontées conséquentes, également appelées contributions, sont faites régulièrement envers la communauté.

L'Éof fait la promotion du logiciel libre à travers l'enseignement. Pour parvenir à cela, ses cours sont pour la plupart publiés au fur et mesure de leur création, et sont accessibles librement et gratuitement. Le logiciel de conception de formation ainsi que la plate-forme d'enseignement à distance sont disponibles sous licence GPL. Tout le bénéfice perçu à travers les inscriptions est reversé à la fondation Éof qui distribue les fonds à des associations ou projets du libre.

III La formation à distance

La formation à distance, ou le mot consacré *e-learning*, est une méthode très en vogue, car elle permet de supprimer les barrières physiques (locaux, place, équipement…) et beaucoup d'organismes publics et privés s'y essayent avec plus ou moins de succès.

En effet, un bon professeur en présentiel, c'est à dire dans un même lieu physique que ses élèves doit savoir capter son auditoire et être un bon pédagogue, au-delà de la connaissance intrinsèque de ce qu'il enseigne. Dès lors que l'on passe sur un apprentissage à distance, les critères que nous venons de définir ne sont plus suffisants. En outre, la maîtrise des outils de communication, l'habitude de la dématérialisation et la communication principalement écrite deviennent des facteurs primordiaux dans cet enseignement. La scénarisation des séquences, le temps d'étude, le type de cours sont également des facteurs à prendre en considération avant même le début de la formation.

Si nombre d'écoles utilisent uniquement la formation à distance comme un moyen d'enseignement, l'Éof place également la distance comme un objectif pédagogique : l'élève doit être en mesure de pouvoir travailler sur des projets en relation avec des personnes éloignées géographiquement.

Pour cela, l'École a travaillé en relation avec le département Innovation Pédagogique de l'ENST Paris, ce qui a donné pour résultat la naissance d'Amarante [1], plate-forme de formation ouverte à distance,

[1] http://www.amarante.eu.org

qui permet un suivi des utilisateurs selon un calendrier établi, avec différents découpages : avancement des apprenants, répartition par type d'activité, etc.

La dématérialisation agit également sur le suivi des élèves. Ainsi, les notions de contrôles écrits et oraux n'ont plus place. On peut difficilement faire passer un QCM en s'assurant que l'étudiant remplit seul son formulaire. Pour pallier cette difficulté, l'Éof a créé un nouveau système d'évaluation qui se base sur les critères suivants :

- efforts effectués sur le parcours technique,
- comportement dans la relation avec l'enseignant, avec le groupe,
- formulation des questions,
- partage et explication des problèmes,
- intégration de l'état d'esprit du libre.

L'écart existant entre un élève possédant une moyenne de 9,5 et un autre de 10 pour obtenir une validation étant faible, les critères sont moyennés par des échelles qui, au final, permettent aux professeurs de savoir si l'élève a plutôt réussi son cursus ou non.

IV L'avenir

De nouveaux partenariats voient le jour (Free, CCI, sociétés de service…) et l'offre de formation s'avère intéressante pour un grand nombre de personnes. L'école espère donc poursuivre son activité en consolidant son cursus professionnel et en intégrant de nouveaux formateurs et formations.

V Le mémoire

C'est de manière naturelle que ce mémoire s'est articulé autour d'un travail à distance, en utilisant les nouvelles technologies comme le courrier électronique, la messagerie instantanée. De nombreuses technologies sont abordées durant cette étude. La plupart prennent leurs fondements sur des standards ouverts, telles que les *RFC*s que l'on peut assimiler à une liste de standards de protocoles informatiques ; leur disponibilité en ligne et leur gratuité autorisent les commentaires de nombreux contributeurs, mais également la possibilité d'une implantation tant des particuliers que des grandes entreprises.

La contrainte du logiciel libre dans la réalisation de ce mémoire est avant tout une force, puisqu'elle autorise la prise de connaissance en profondeur des logiciels qui seront employés, c'est à dire leur code source, ce qui permet de vérifier ou faire vérifier la validité des implémentations, éviter les vices cachés (backdoor, malware, spyware…).

Ce mémoire doit à la fois permettre l'étude d'une plate-forme complète, mais également donner la possibilité d'études séparées, afin de pouvoir servir comme ressource pédagogique pour un public technique.

La plate-forme que nous décrivons est administrée par deux personnes au minimum, afin d'assurer un suivi opérationnel tout au long de l'année. Pour cela, nous avons bénéficié des structures déjà effectives au sein du service informatique de l'école :

- un wiki, logiciel web permettant le travail collaboratif sur des textes et documents. Chaque configuration peut être vue et discutée. La documentation de support (arrêt et relance des applicatifs, processus de mise à jour…) est décrite dans les pages du wiki ;
- une liste de discussion, qui permet les échanges à vif, sur des décisions, comme les montées de version, la réaction face à une tentative de piratage ou encore les problèmes rencontrés ou

les technologies que l'on envisage de déployer. Cette liste possède une clé cryptographique qui permet l'envoi de messages chiffrés à tous les abonnés (ici, les administrateurs) ;
– un carnet de bord qui remonte toutes les mises à jour et les astuces de l'administration au quotidien.

Chapitre 2

Architecture de services

Ce chapitre nous permet de découvrir la maquette qui servira de base à nos expériences. Dans un premier temps, nous aborderons la description des services serveurs, puis, avec une vision plus fonctionnelle, nous décrirons les interactions entre les différents services.

I L'architecture physique

La contrainte forte énoncée par le sujet nous rappelle que tout service doit être délivré en s'appuyant sur des logiciels libres. Ainsi, même lorsque des outils prêts à l'emploi sont disponibles sur le marché, nous préférerons déployer une technologie libre.

Sur ce premier schéma 2.1, nous retrouvons l'ensemble des composants de notre architecture :

- un pare-feu, basé sur le couple Netfilter/Iptables,
- un répartiteur de charge,
- un serveur virtuel utilisant le patch *Linux Vservers*, hébergeant différents services web et dns,
- les services web portés par des serveurs *apache*,
- les *dns* proposés par *Bind*,
- un serveur syslog,
- et un pot de miel (en anglais, *honeypot*).

Chaque serveur décrit ci-dessus est installé sur une distribution Debian GNU/Linux (version etch) dont le noyau a été recompilé pour répondre au mieux à nos besoins. Il comporte ainsi un minimum de périphériques supportés (pas d'IRDA, bluetooth, wifi, carte d'acquisition, ...), une restriction au niveau des protocoles (NFS, samba, ...).

II Matérialisation des flux réseaux

Le schéma 2.2 représente la vision logique de notre architecture. Nous décrivons ici comment communiquent les différents éléments qui la composent ainsi que la manière d'accéder à nos serveurs. Nous considérons que l'administration de la structure se fait à distance. Ainsi, nous accéderons au travers du nuage Internet, comme un visiteur normal. Une alternative possible pour minimiser les risques serait de monter un tunnel *VPN* entre les administrateurs du site et le site lui-même. Pour des raisons de temps, cette possibilité ne sera pas abordée, elle reste cependant toutefois une bonne alternative technique.

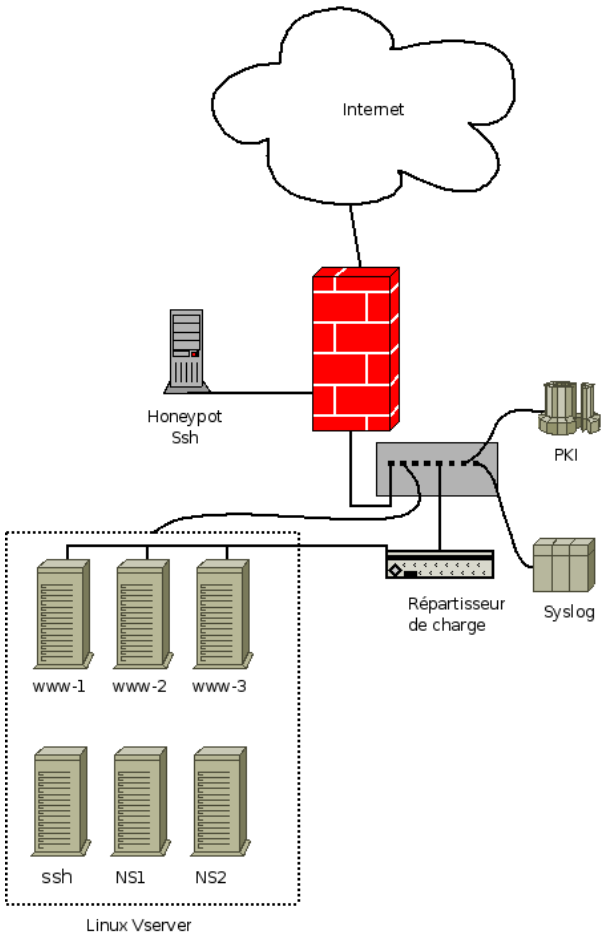


FIG. 2.1 – Architecture physique du réseau

Commençons l'analyse des flux proposés. Pour cela, nous regardons en premier lieu l'accès *ssh* (flux 1) qui permet la connexion et l'administration distante (chap. 7). Nous pouvons distinguer deux types de flux différents. Le premier, ici en couleur orange, indique un flux d'administration normal. On constate que seul l'accès de l'extérieur vers le *Vserver* est autorisé. En effet, une machine virtuelle (chap. 3) est implémentée pour servir uniquement de rebond vers les autres serveurs. Cela permet de limiter le nombre de serveurs à l'écoute sur le port 22 vers Internet, mais également d'identifier toute personne qui tenterait une connexion illégitime vers une de nos machines.

L'autre flux *ssh* (flux 2) est, lui, uniquement dédié au pot de miel. En effet, les actions analysées et les faiblesses portent sur ce service. Ainsi, tout pirate ou robot qui tentera de se connecter sur cette machine sera automatiquement repéré et ses actions pourront être analysées (chap. 12).

L'interrogation *DNS* (flux4) passe elle aussi par l'intermédiaire de notre firewall, pour toute personne souhaitant connaître l'adresse *IP* correspondant à un nom ou service de notre domaine. La réplication DNS (flux 5) permet également la redondance de l'information et la disponibilité en cas de perte d'un serveur (chap. 4).

La connexion en *HTTP* (flux 3) sur les serveurs se fait en deux étapes. la première étape consiste à l'envoi de toutes les requêtes vers le répartiteur de charge (chap. 6) qui, en deuxième étape, distribue les requêtes vers les trois serveurs Internet présents. Ces accès se feront par l'intermédiaire du port 80 (http) ou du port 443 (https) (chap. 5). La réplication proposée permet une disponibilité supplémentaire en cas de perte de l'un des serveurs, celle-ci opérant par l'intermédiaire du service *rsync* (flux 6).

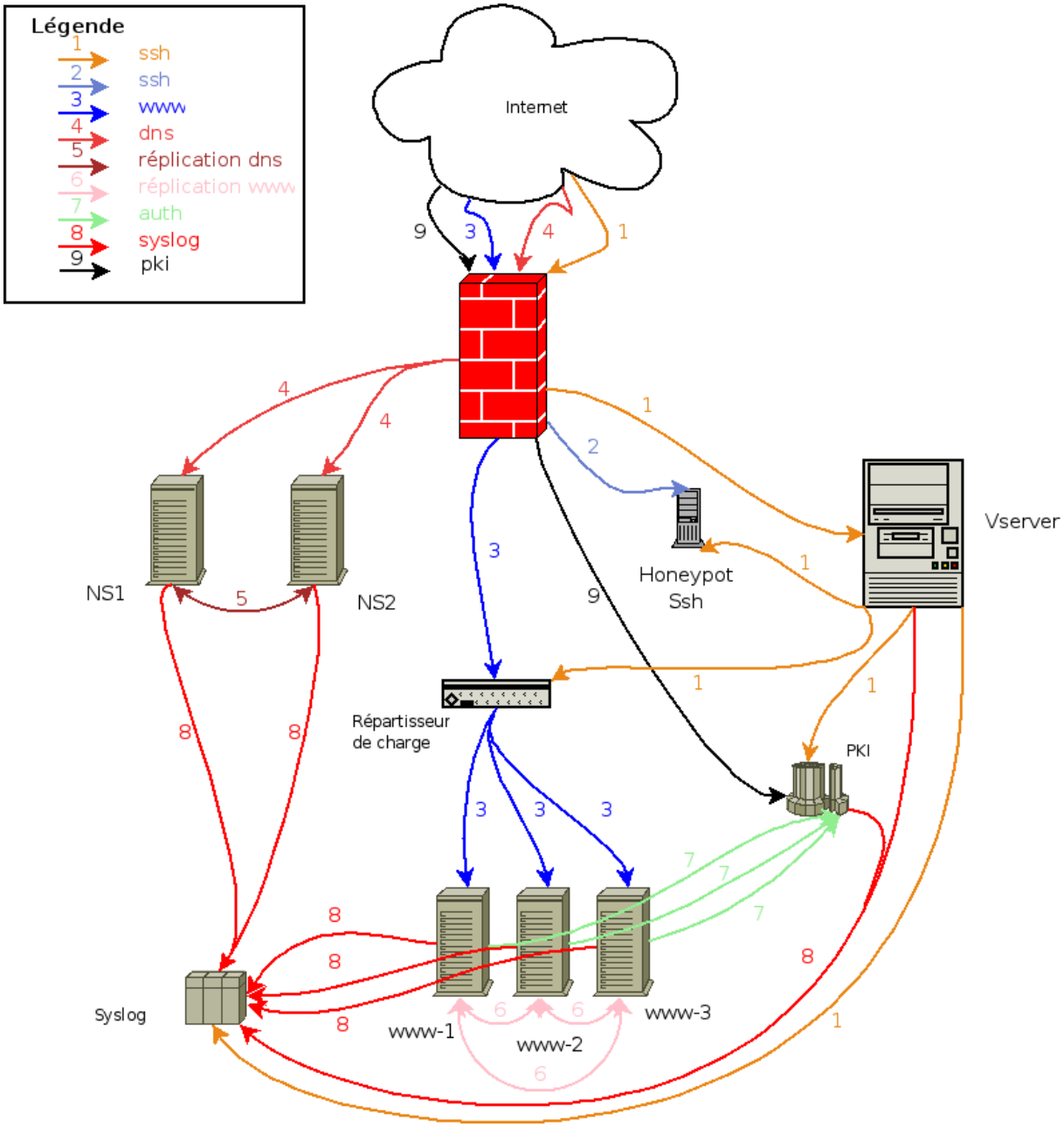


FIG. 2.2 – Schéma des flux réseaux

Les serveurs web proposent un ensemble de sites web nécessitant une authentification. Pour ce faire, une liaison avec le serveur *PKI*, présenté sous l'appellation *auth* (flux 7) sur le schéma est nécessaire (chap. 11). La connexion à la PKI s'effectue par l'intermédiaire d'un client spécifique (flux 9). Ce flux peut également être encapsulé dans un tunnel SSH afin de minimiser le type de flux entrants sur notre réseau.

L'ensemble des serveurs corrèlent des événements, des informations sur leur fonctionnement. Afin de pouvoir les rassembler et les analyser, nous utiliserons un serveur Syslog (chap. 9) qui aura cette fonction. Tous les serveurs doivent par conséquent remonter un flux d'information (flux 8) vers le serveur syslog. Ce serveur est critique en cas de débuggage ou de piratage, puisqu'il reste l'unique point de sauvegarde de l'infrastructure.

La détection d'intrusion est le dernier élément qui vient compléter notre puzzle. C'est un élément réparti qui prendra plusieurs formes : soit un contrôleur d'intégrité système (chap. 8), pour assurer la

cohérence des binaires ainsi que de la non modification par un tiers, soit un piège à scanner (chap. 10), afin de reconduire toute tentative à la porte de notre réseau, ou encore un gardien de l'authentification (chap. 7), en interdisant l'accès à toute personne commettant de nombreuses tentatives d'authentification.

Chapitre 3

Virtualisation

I Principes de virtualisation

1. Histoire

C'est en France, dans les laboratoires d'IBM, à Grenoble, que les premiers travaux sur les machines virtuelles font leur apparition. Nous sommes alors en 1960 et Big Blue vient de créer son premier ordinateur avec une base logicielle commune, le *System/360*, qui pour la première fois n'obligera pas à changer tout le matériel (imprimante, carte...) lors de la prochaine évolution du processeur. Cela permettra également de conserver les programmes, sans avoir à en ré-écrire spécifiquement des parties. Cette première évolution sera continuée par la naissance du *CP-40* [2], considérée comme la première machine virtuelle [3]. Le produit se transforma en *VM/CMS*. La suite logique donnée par IBM les conduira vers les *mainframe* qui virtualisent leur système d'exploitation. IBM étant à l'époque un vendeur de matériel, il ne pouvait alors pas faire davantage la promotion d'une technologie qui aurait permis au client d'économiser sur ses ventes.

Par la suite, d'autres sociétés comme HP, avec ses *PA-RISC ou IA64*, ou encore Sun avec la série des *E10K/E15K/E20K/E25K* utiliseront la technologie de virtualisation.

Le système commercial le plus connu actuellement se nomme *VMware*, du même nom que la société qui le commercialise. C'est cette société qui popularisera auprès du grand public l'intérêt des machines virtuelles, que ce soit pour des tests ou de la production. Cependant, bien que la société VMware soit leader dans le domaine, elle privilégie l'usage des machines virtuelles en tant que machine virtuelle, ce qui ne représente qu'un des types de la virtualisation [4] [5], comme nous le verrons dans la section II. De nombreux logiciels libres existent aujourd'hui à tous les niveaux logiciels, nous en citeront quelques uns pour nous attarder plus particulièrement sur *Linux Vservers*.

2. Usages

Tout comme nous l'avons vu, l'usage premier n'est pas récent et concernait les grands constructeurs de matériels et de systèmes. C'est cependant avec l'explosion d'Internet que la technologie de virtualisation verra de nouveaux utilisateurs. En effet, les premiers *ordinateurs* personnels grand public, comme ceux d'Atari et Amstrad verront une communauté intéressée à retrouver leurs anciens programmes ou jeux. Il en va de même pour les émulateurs de console de jeux.

Un domaine davantage dédié à la recherche, la sécurité informatique, utilise beaucoup ce principe de virtualisation. Celui des pots de miel, qui, comme nous le verrons dans le chapitre dédié à ce sujet, permet aux administrateurs d'émuler tout ou partie d'un système sans devoir compromettre les *hôtes*, à savoir la première couche système.

C'est plus récemment que le monde de la production revient à cette notion, par besoin de rationalisation des coûts et des usages. Ainsi, le développement et la mise en production sont facilités par le seul déplacement des machines virtuelles. Des tests peuvent dès lors être effectués sans avoir peur de casser le système hôte. Les nouveaux systèmes sont de simples copies logicielles des systèmes de base. En terme de sécurité, toute attaque peut être limitée à l'hôte virtualisé, rendant de ce fait l'attaque dans un environnement clos. Pour finir, une meilleure gestion de la puissance des processeurs et de la mémoire peut être envisagée.

II Les différentes familles de virtualiseurs

La virtualisation peut se présenter sous différentes formes, que ce soit de manière matérielle, par une émulation de machine complète, le partage de ressources via un isolateur, ou encore des applications virtuelles, tout comme le permet la technologie *java* avec sa fameuse machine virtuelle. Nous présentons dans cette section les différents types de virtualisation qui étaient à notre disposition et ce qui a orienté le choix de la solution.

1. Théorie

Avant de découvrir les différents types de virtualisation qui existent, il est nécessaire d'expliquer les espaces d'exécutions, communément appelés *ring*. Les rings ou anneaux sont des espaces protégés, mis à disposition par le processeur. Le niveau de sécurité s'étend du plus sécurisé, appelé le *Ring0* au plus permissif *Ring3*, tout comme nous le montre la figure 3.1.

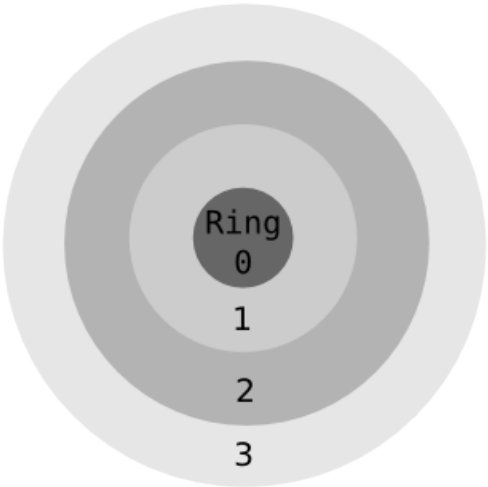


FIG. 3.1 – Concept des anneaux

De ce fait, lorsqu'une application se trouve par exemple dans le *Ring3*, elle doit faire appel à une instruction spéciale pour pouvoir utiliser un périphérique qui se trouverait dans une couche inférieure. Typiquement, les processeurs de type X86 possèdent quatre anneaux. La plupart des systèmes fonction-

nant sur ces processeurs n'emploient que deux de ces anneaux, le Ring0 appelé *kernel land* et le Ring3 nommé *user land*.

2. Matériel

La généralisation des logiciels de virtualisation répond à un besoin, mais aussi à une demande. Ces technologies, incluses directement au coeur des processeurs, sont axées sur l'administrabilité et la sécurisation. Pour la première, c'est la gestion de plusieurs couches logicielles, présentées sur différentes partitions, permettant ainsi, comme au niveau logiciel, d'offrir plusieurs *machines*, de prévoir des tests, des bascules. La sécurisation se porte au niveau des piles mémoires qui peuvent être mieux gérées et mieux contrôlées, indépendemment du système d'exploitation. Les acteurs majeurs dans ce domaine sont *INTEL* avec son *Vanderpool* [6] et *AMD* avec *PACIFICA*. Cette technologie est cependant encore très jeune et a déjà été prise en défaut par le chercheur en sécurité, Joanna Rutkowska [7].

3. Isolateur

L'isolation est issue du procédé de la mise en place d'un niveau de virtualisation du système d'exploitation. Ainsi, le noyau, la mémoire, le CPU peuvent être partagés. L'application lancée dans un *contexte*, également appelé zone d'exécution, ne *connaîtra* que les applications lancées dans le même contexte.

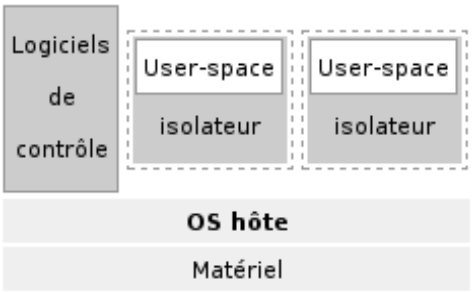


FIG. 3.2 – Architecture d'un isolateur

L'isolateur se présente sous la forme d'un patch pour le noyau et gère le partage de la mémoire, du CPU et s'assure qu'un utilisateur ou une application ne pourra pas *sauter* d'un contexte vers un autre contexte. En ce sens, l'utilitaire *chroot* ne peut pas être considéré comme un isolateur, puisque son action consiste uniquement à changer le chemin de la racine. C'est ce qu'illustre Brad Spengler dans son article [8] sur les différentes façons de briser la *prison de verre* et la solution fournie par son patch destiné au noyau, GrSec [1]. Le schéma 3.2 montre la place de l'isolateur au sein du système, avec l'exécution en *userspace* des applications et la présence d'un logiciel de contrôle. Les produits disponibles sur le marché sont *Linux-Vserver*, *BSD Jails* ou *OpenVZ*.

4. Noyau en espace utilisateur

Il est possible de considérer un noyau comme une application. C'est la solution de virtualisation proposée par exemple par *User Mode Linux* ou encore *Cooperative Linux*. Dès lors, l'accès au matériel

[1] http://www.grsecurity.org

ne s'effectue plus que par un dialogue avec le noyau de l'hôte. Cette solution est donc assez restrictive en terme de capacité d'interaction puisque le noyau n'est alors plus utile que pour la gestion des applications qui seront *au-dessus* de lui.

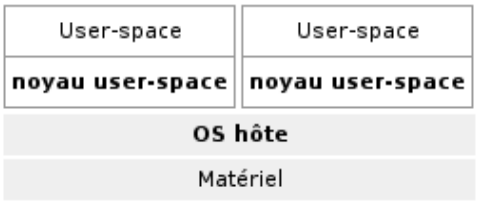


FIG. 3.3 – Architecture dans l'espace utilisateur

Il est préférable d'utiliser cette solution dans le cadre de développement de nouveaux noyaux, ou alors pour installer un système d'exploitation dans un environnement à droit restreint, où l'utilisateur pourrait installer des applications, mais pas supprimer le système existant.

5. Machine Virtuelle

La machine virtuelle est sûrement le mode de virtualisation le plus connu, puisque popularisé auprès du grand public par l'intermédiaire de la société *VMware*. L'idée avancée est de reconstituer un environnement complet, y compris matériel et de mettre à disposition une plate-forme complète de virtualisation, capable de supporter différents environnements simultanément.

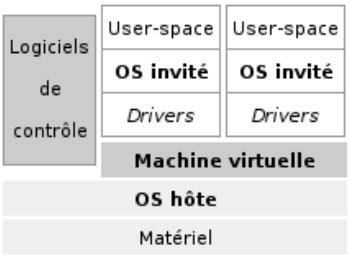


FIG. 3.4 – Architecture d'une machine virtuelle

Bien que cette solution soit plébiscitée du public, elle demande néanmoins de grosses configurations matérielles afin de supporter la reconstruction matérielle, en surcouche d'un système d'exploitation, déjà consommateur de ressources. Les logiciels libres connus pour cet usage sont *QEMU* et *VirtualBox*.

6. Hyperviseur

L'hyperviseur est une couche directement posée au-dessus du matériel. C'est un noyau optimisé pour la gestion directe de systèmes d'exploitation. Ainsi, contrairement aux autres méthodes, il n'y a pas de latence due à un système hôte. Les surcouches sont elles aussi, normalement, optimisées pour fonctionner avec l'hyperviseur. Les résultats sont donc optimaux. Le logiciel *XEN* est à ce jour le seul produit dans ce domaine, bien que de nouveaux concurrents apparaissent.

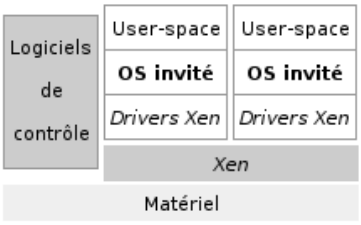


FIG. 3.5 – Architecture d'un hyperviseur

7. Choix de la machine virtuelle

Comme nous l'avons déjà expliqué, notre travail se base sur un choix de logiciels libres. De ce fait, nous pouvions nous porter sur les technologies et logiciels suivants :

- isolateur avec *Linux-Vserver*,
- espace utilisateur avec *User Mode Linux*,
- machine virtuelle avec *VirtualBox* ou *Qemu*,
- hyperviseur avec *Xen*.

Chacune des technologies a ses avantages et ses inconvénients. Au-delà de nos besoins, nous pouvons d'ores et déjà regarder les résultats obtenus par *Quétier* dans son étude sur la virtualisation [9]. Les résultats obtenus pour *VMware* peuvent être considérés identiques pour *VirtualBox*, la technologie étant similaire.

TAB. 3.1 – Comparaison des temps de lancement à froid et à chaud de 4 outils de virtualisation

	Lancement à froid	Lancement à chaud
Vserver (100 VMs)	460s	30s
UML (100 VMs)	1040s	160s
Xen (50 VMs)	860s	860s
VMware (50 VMs)	1400s	810s

Ce premier tableau 3.1 montre des différences lors de la première initialisation, appelée lancement à froid, puis lors de démarrage alors que les bibliothèques sont déjà en mémoire. L'étude sur *Xen* et *VMware* se limite à 50 instances, limitation de la gestion des IRQ par le matériel. On constate que l'isolateur démarre plus rapidement que les autres outils et gagne 1500% de temps de chargement lorsque ses bibliothèques sont déjà présentes.

Les expériences relatées dans les graphiques 3.6 et 3.7 se fondent sur l'observation de la charge issue de l'utilisation d'une application au sein d'un outil de virtualisation en comparaison avec la charge de la même application sur un hôte nu. Le premier graphique 3.6 montre très clairement que la technologie de machine virtuelle utilise énormément le CPU avec une courbe qui tend vers une exponentielle. L'isolateur et l'hyperviseur restent très proches de la charge théorique.

Concernant la surcharge mémoire, la figure 3.7 montre un comportement *erratique* de l'outil UML, alors que les résultats de l'isolateur et l'hyperviseur restent toujours aussi proche de la théorie. Tout comme lors de l'analyse du CPU, l'application dans la machine virtuelle consomme d'autant de mémoire qu'il y a de machines virtuelles lancées.

En tirant parti de ces résultats, mais également des moyens mis à notre disposition, les logiciels Xen et Linux-Vserver semblent convenir à nos besoins. Cependant, il s'avère inutile d'avoir des noyaux

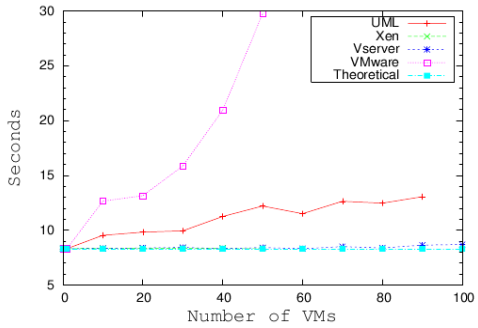


FIG. 3.6 – Charge du CPU en fonction du nombre de VM

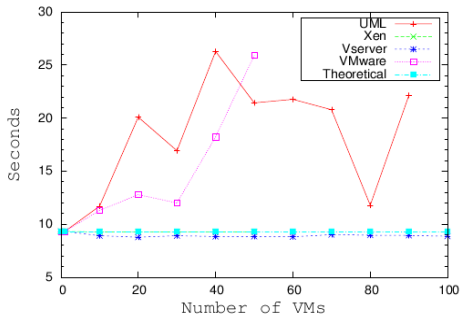


FIG. 3.7 – Charge de la mémoire en fonction du nombre de VM

spécifiques, inutile également d'avoir accès à différents systèmes d'exploitation, puisque nous utiliserons uniquement GNU/Linux. L'usage requiert uniquement une séparation et de la sécurité entre les différentes machines ou contextes. Pour tout cela, notre choix se porte sur Linux-Vserver qui s'avère également plus simple dans son administration et moins gourmand en espace disque.

III Linux-Vserver

1. Fonctionnement

Tout serveur se constitue de trois éléments essentiels :
- le matériel,
- le noyau,
- les applications.

Le matériel peut être spécifique selon les besoins de service. Le noyau joue alors le rôle d'interface, de couche d'abstraction, donnant accès à des applications qui n'ont pas besoin de connaître les spécificités du matériel pour pouvoir communiquer avec lui. Certaines applications, quant à elles, nécessitent une instance père unique pour pouvoir fonctionner. De ce fait, tout démarrage d'une autre occurrence de cette entité serait vouée à l'échec. La mise en place de Serveurs Privés Virtuels (*VPS*) permet donc d'isoler la couche applicative en donnant l'accès à un seul et unique noyau qui se charge de la communication avec le matériel.

Cette solution permet à la fois de spécifier des adresses précises et différentes pour chaque VPS, mais également d'avoir un hôte central en mesure de gérer et d'accéder à tous les contextes hébergés.

Pour finir, n'importe quelle distribution GNU/Linux peut être installée, indépendamment du noyau. La liberté d'utilisation est donc conservée et permet de s'adapter aux connaissances de chaque utilisateur.

2. Installation de Linux-Vserver

Linux-Vserver est disponible sur douze architectures matérielles différentes, dont celle qui nous intéresse, la plate-forme x86. Pour pouvoir l'installer, nous avons besoin de deux choses : le patch pour le noyau linux et les outils de gestion de l'isolateur.

Nous allons commencer par la récupération et l'installation du patch. Pour ce faire, il nous suffit de nous connecter au site internet et de récupérer un fichier *diff* compressé, qui comporte les modifications à apporter au noyau. Nous nous positionnons donc dans le répertoire des sources du noyau pour télécharger le fichier :

```
cd /usr/src/linux
wget http://ftp.linux-vserver.org/pub/kernel/\
vs2.0/patch-2.6.17.14-grsec2.1.9-vs2.0.2.1.diff.bz2
```

Lorsque le téléchargement est terminé, nous pouvons appliquer les modifications au noyau afin d'insérer les routines nécessaires au fonctionnement de Vserver :

```
bzcat ./patch-2.6.17.14-grsec2.1.9-vs2.0.2.1.diff.bz2 | patch -p1
```

Le noyau est à présent prêt à être configuré. On se reportera au chapitre VII pour la compilation. Il faut à présent installer les outils qui vont nous permettre de manipuler nos contextes :

```
apt-get install util-vserver vserver-debiantools
```

Le premier paquet contient un ensemble de binaires qui permettent entre autres de démarrer les contextes et d'y entrer. Le deuxième livre le script *newvserver* qui permet la création de contextes.

3. Configuration de Linux-Vserver

Il existe deux étapes de configuration. La première est obligatoire, puisque c'est le choix des options à activer au niveau du noyau. La deuxième concerne l'ajustement des contextes installés. Cette deuxième étape n'intervient que si l'on souhaite étendre les montages, changer les adresses IP…Notre usage ne nécessite pas d'affinement particulier, nous utiliserons donc la configuration par défaut, tout en nous laissant la possibilité d'y revenir ultérieurement.

Voici les options que nous activons au niveau du noyau :

```
#Linux Vserver
CONFIG_VSERVER_PROC_SECURE=y
```

Ceci configure la sécurité *ProcFS* pour cacher dès l'initialisation les entrées non-process pour tous les contextes excepté le principal et spectateur qui est sécurisé par défaut.

```
CONFIG_VSERVER_HARDCPU=y
```

Cette option permet à l'ordonnanceur du panier de jetons de suspendre un processus lorsque les jetons d'un contexte sont vides. Les processus appartenant à ce contexte ne seront plus en mesure d'utiliser des ressources CPU jusqu'à ce qu'un nombre de jetons minimum par contexte soit atteint.

```
CONFIG_VSERVER_HARDCPU_IDLE=y
```

Mise en place de la limitation des tranches d'inactivité dans le CPU. De ce fait, le prochain contexte pourra être planifié le plus tôt possible.

```
CONFIG_INOXID_UGID24=y
```

L'utilisation de *Quota* ou de *Disk Limits* nécessite l'option de tagging. Cette configuration ajoute l'information de contexte persistant aux systèmes de fichiers montés avec l'option *tagxid*.

Il ne reste plus qu'à compiler le noyau et l'installer (voir le chapitre 8).

4. Mise en production

Lorsque le nouveau noyau fonctionne, nous pouvons créer nos environnements. Pour cela, il existe deux méthodes. Soit nous utilisons le script *newserver* fournit par *Debian*,

```
newvserver --vsroot /var/lib/vservers/ --hostname vserver1 \
--domain eof.eu.org --ip 192.168.0.10/24 \
--dist etch --mirror http://ftp.fr.debian.org/debian/
```

soit nous utilisons la commande générique *vserver*.

```
vserver vserver2 build  -n vserver2   --hostname vserver2.eof.eu.org
--i eth0:192.168.0.11/24   -m debootstrap -- -d sarge
```

Dans les deux cas, nous construisons ici un environnement vierge, à partir de paquets que nous allons chercher sur Internet, pour former des contextes issus de la distribution GNU/Debian. L'adresse IP est affectée depuis l'initialisation.

Lorsque l'opération est achevée, il ne reste plus qu'à démarrer notre nouveau contexte :

```
vserver vserver1 start
```

Notre vserver est maintenant fonctionnel, nous pouvons désormais le mettre à disposition ou l'utiliser pour héberger des applications.

Si nous souhaitons joindre le nouveau contexte depuis la machine hôte, il suffit *d'entrer* dans celui-ci :

```
vserver vserver1 enter
```

De même, depuis l'hôte, nous pouvons avoir les renseignements sur les contextes qui sont lancés à l'aide de la commande *vtop* :

```
~# vtop
CTX   PROC    VSZ    RSS  userTIME   sysTIME    UPTIME NAME
0       53 345.7M  96.7M   0m38s80   0m11s83   3h30m05 root server
49153    6  53.1M   6.5M   0m00s70   0m00s20   2m18s89 vserver1
49154    5  45.4M   4.6M   0m00s10   0m00s30   0m05s80 vserver2
```

avec pour chaque entête de colonne les définitions suivantes :

- **CTX** : numéro du contexte. 0 pour le root contexte, 1 pour le contexte de supervision,
- **PROC QTY** : nombre de processus dans chaque contexte,
- **VSZ** : nombre de pages virtuelles de mémoire,
- **RSS** : taille résidente,
- **utime** : temps CPU User-mode accumulé,
- **ctime** : temps CPU Kernel-mode accumulé.

Lors de la mise en place de notre maquette, nous avons utilisé nos contextes pour héberger des serveurs apache, en liaison avec le répartiteur de charge, mais nous avons également testé la mise en place du pot de miel kojoney (voir le chapitre 12, page 101). Pour la mise en place des serveurs DNS, il nous a fallu recompiler les paquets fournis par Debian, car ceux-ci n'avaient pas les *capabilities* activées par défaut, ce qui empêchait le lancement des services serveurs DNS.

Chapitre 4

DNS

I Histoire des DNS

À la préhistoire d'Internet, nous le savons, le projet Arpanet demandé par le DoD (Department Of Defense, États-Unis) devait connecter des sites primordiaux et s'assurer qu'en cas de perte d'un site, la connectivité permettrait au reste du réseau de continuer à communiquer. Dans cet environnement, un fichier unique, nommé *HOSTS.TXT*, conservait la correspondance de chaque nom et adresse IP des machines. Chaque Unix connecté à Arpanet générait son propre fichier */etc/hosts* à partir de ce fichier central. C'était là une des épines dorsales d'Arpanet. Ce fichier était maintenu par le Network Information Center (NIC) au sein du Stanford Research Institute (SRI). La procédure de l'époque était assez simple. Le NIC recevait par mail les modifications qu'il devait apporter au fichier central. Il reportait ces modifications sur le fichier, puis positionnait la nouvelle mouture sur un ftp, une à deux fois par semaine, que tout un chacun pouvait dès lors récupérer, afin de tenir à jour son registre.

Au début des années 1980, Arpanet s'est transformé pour adopter le protocole développé par Vinton Cerf [10], *TCP/IP*, ce qui engendra une croissance forte en terme de machines connectées. Ceci eu pour effet de rendre obsolète le mécanisme en vigueur pour la mise à jour des *DNS*, exposé aux problèmes suivants :

- la charge réseau générée par les hôtes toujours en augmentation,
- la charge processeur de la machine hôte centrale,
- la perte d'intégrité sur l'unicité des noms de machine présents dans le fichier,
- risque de déni de service par homonymie,
- la cohérence de temps entre la mise à jour du fichier et la diffusion sur l'ensemble du réseau.

C'est un ingénieur, Paul Mockapetris [11], qui résolut ce problème en concevant l'architecture des DNS que nous connaissons aujourd'hui. Il créa le premier serveur DNS [12], nommé Jeeves, et transcrivit son travail au sein des RFC 882 et 883. Quelque temps plus tard, celles-ci furent complétées par les RFC 1034 et 1035 [13] qui sont encore aujourd'hui les standards de base de cette technologie.

Par la suite, quatre étudiants de l'Université de Berkeley travaillèrent avec le groupe de recherche chargé d'ARPAnet sur un logiciel permettant de répondre à ces recommandations. Le fruit de leur travail se nomme *Bind* [14] qui est devenu le standard de facto en matière de serveur dns. Par la suite, c'est Paul Vixie [15], à l'époque employé de DEC qui prend le relais en 1988 et fondera, six ans plus tard l'*Internet Software Consortium* (ISC), organisation à but non lucratif, et qui, jusqu'à ce jour, a la charge du développement et du déploiement de Bind. L'organisme a changé de nom en 2004 pour s'appeler *Internet Systems Consortium*.

Ce travail tira les conséquences de l'architecture fondée sur le fichier unique *HOSTS.TXT* et se rapprocha le plus possible d'un système qui avait pu faire preuve de sa robustesse au fur et à mesure des années. C'est ainsi que Mockapetris calqua son travail sur une arborescence de système *Unix*.

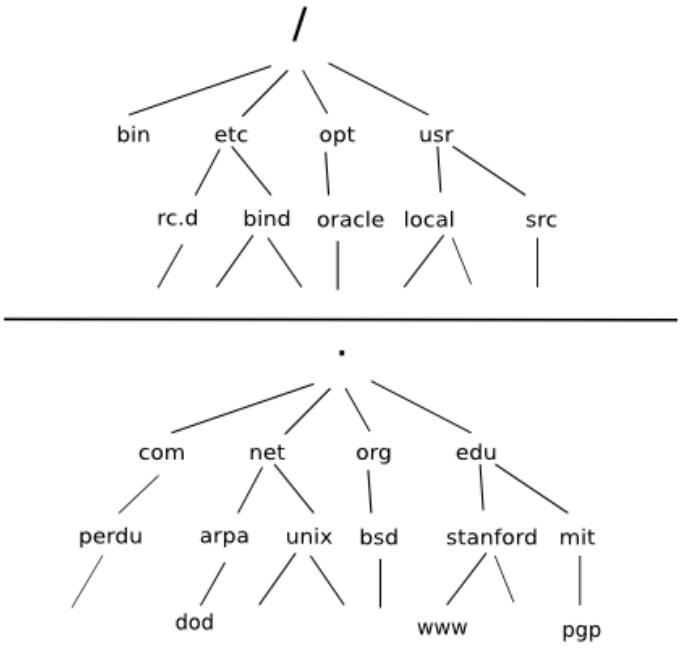


FIG. 4.1 – Architecture du Unix vs DNS

La racine prend ici la forme d'un point (.). Chaque noeud de l'arbre devient le père des noeuds présents en dessous de lui, tout comme un dossier serait le parent des sous-dossiers qui lui sont affiliés. L'ensemble ainsi formé permet d'obtenir une cohérence sur le caractère unique de chaque nom de domaine. Pour obtenir le nom (ou chemin complet) d'un dns, on partira de la branche la plus basse et on remontera jusqu'à la racine.

Si nous prenons exemple sur le schéma précédent 4.1, nous obtenons la chaîne suivante : `www` ⇒ `stanford` ⇒ `edu` ⇒ `.`

Chaque *noeud* de l'arbre est associé par un point (.).

Ce système de nommage permet ainsi de dédier l'administration d'un noeud à un autre domaine. Ce n'est donc plus un site qui centralise les informations, mais chaque site référence ses propres informations ainsi que celles de ses sous-domaines. Dans notre exemple, le domaine stanford.edu est délégué à l'Université de Stanford.

II Espace de nommage

Cet arbre inversé qui contient les noms indexés est appelé espace de nommage. Le haut de l'arbre représenté par un point est appelé la racine et chacun des noms est appelé noeud. L'arbre se limite à 127 niveaux. Chaque noeud est limité à 63 caractères. Pour pouvoir lire correctement un nom de domaine à partir de l'arbre DNS, il faut le lire de droite à gauche et de bas en haut, du noeud vers la racine. Chaque noeud fils est obligatoirement unique et ne peut avoir de doublon sous un même noeud père.

Pris séparément, un arbre est constitué de sous-arbres dont chaque sommet est le noeud le plus

élevé (soit le plus proche de la racine) ; de fait un noeud peut appartenir à plusieurs domaines. Le trigramme *www* du nom *www.eof.eu.org* appartient au domaine *eof*, qui est lui-même inclus sous le domaine *eu* qui est sous la désignation du domaine *org*.

Lorsque l'on termine le domaine par le *point* racine, on parle alors de nom totalement qualifié (ou *Fully Qualified Domain Name* abrévié *FQDN*). Voici un exemple de FQDN :

```
www.eof.eu.org
```

Les domaines directement reliés à la racine sont appelés des domaines de premier niveau (*Top Level Domain*) comme *.net* ou *.com*, les domaines de second niveau étant issus des domaines de premier niveau (*secondary level domain*), comme *exemple.com*.

III Espace des noms du domaine Internet

1. Domaines de niveau supérieur

Voici la liste des domaines de niveau supérieur, c'est à dire les extensions finales des noms de domaine. Cette liste est validée par l'ICANN[1].

- aero : industrie du transport aérien
- biz : domaine des affaires (business)
- cat : langue et communauté catalane
- com : générique, extension de commercial
- coop : association coopérative
- info : générique, extension d'information
- jobs : gestion des ressources humaines
- mobi : clients et fournisseurs de produits et services mobiles
- museum : musée
- name : individu
- net : générique, extension pour les organismes travaillant sur le réseau
- org : générique, extension pour les organisations non-commerciales
- pro : professionnels
- travel : activité du voyage
- gov : organisation gouvernementale des États-Unis d'Amérique
- edu : organisme universitaire
- mil : organisation militaire des États-Unis d'Amérique
- int : organisation enregistrée via traités internationaux.

À cette liste déjà longue, il faut ajouter une dénomination par pays, sous forme d'une extension composée de deux caractères, selon le modèle ISO 3166. On pourra aussi souligner la lourde présence des États-Unis d'Amérique. C'est en fait leur antériorité historique en tant que créateur d'ARPAnet qui veut cela. On notera également que le TLD *.us* ne se divise pas en section d'ordre métier, comme on peut le trouver pour le Royaume-Uni, avec *.co.uk* pour les entreprises ou encore *.ac.uk* pour le domaine académique. En effet, le domaine *us* se divise en cinquante sous-domaines, un par état. C'est l'exception à la règle.

[1] Internet Corporation for Assigned Names and Numbers : http ://www.icann.org

IV Délégation d'autorité

Nous l'avons déjà vu, le schéma DNS se dessine en arbre, avec une racine et des noeuds. À la manière d'une administration, il serait fastidieux de gérer la totalité de l'arborescence en un point. De ce fait, tout comme nous avons des administrations décentralisées à différents niveau (état, région, département, ville), l'arbre DNS peut autoriser la délégation de zone à des éléments plus petits. Ainsi, pour le domaine *eof.eu.org*, le domaine *org* délègue la branche *eu* à des gestionnaires, qui nous délèguent à nous, la branche *eof*. Nous avons donc la responsabilité de cette branche, mais nous pouvons également gérer directement le contenu de cette dernière, comme le montre le schéma 4.2.

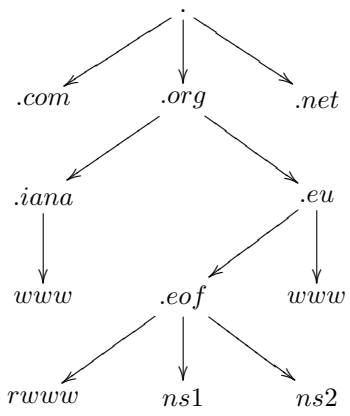


FIG. 4.2 – Délégation de la zone eof.eu.org

V Délégation, domaines et zones

Il est nécessaire de bien comprendre la différence entre un serveur de noms, un domaine et des zones, d'assimiler le lien qui existe entre eux. Un serveur de noms héberge des zones, qui sont elles-même constituées d'enregistrements. Un domaine est lui-même constitué de zones, cependant, un domaine n'est pas forcément hébergé sur un seul serveur de noms. Cela est dû à la capacité de *dns* de pouvoir découper, *déléguer* différentes parties d'un domaine à d'autres serveurs de noms. Ainsi, si nous prenons le cas fictif d'une université française (schéma 4.3), nous pouvons obtenir le découpage suivant :

- l'université `universite` dépend du domaine français `fr`, celui-ci lui délègue l'autorité de gestion de son domaine `universite.fr`.
- l'université se décline en plusieurs composantes telles que :
  - sciences
  - lettres
  - économie
  - droit
  - éducation
- chaque composante peut avoir la possibilité de se voir *déléguer* sa zone.

Chacune de ces parties peut recevoir la délégation de son sous-domaine, qui pourra lui-même être décomposé en plusieurs zones (administration, gestion, enseignement, recherche, ...). De ce fait, les administrateurs des facultés de sciences, lettres et économie sont en mesure de gérer leur sous-domaine et se sont vu déléguer celui-ci. Les zones ne seront donc plus présentes de manière complète sur le serveur du domaine `universite.fr`. Il restera uniquement un pointeur vers les serveurs des zones

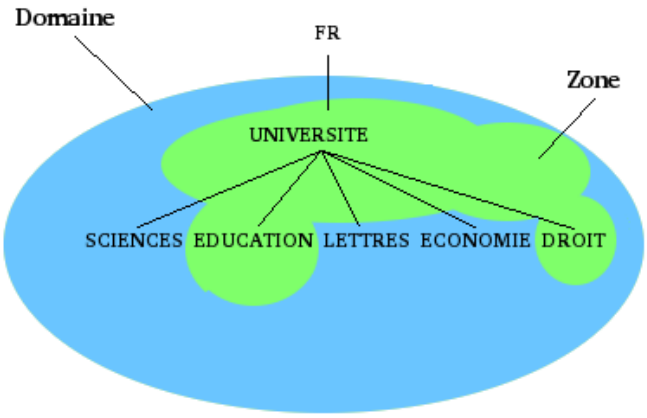


FIG. 4.3 – Domaine, délégation et zone

sciences, lettre et économie. Par contre, aucune délégation n'est effectuée pour les zones droit et éducation, par manque de personnel, par exemple : elles restent donc sur le serveur autoritaire du domaine `universite.fr`. Cette répartition permet ainsi de posséder un serveur de domaine qui ne possède pas l'intégralité des zones de son domaine, mais qui peut en autoriser la délégation. Ce principe est très intéressant pour les gros domaines comme les *Top Level Domain* ou les grandes entités, telles qu'une administration ou une grande entreprise. Dans le cas de l'Éof, ceci n'apporte rien pour l'instant.

VI Résolution de noms

Comme nous l'avons déjà vu précédemment, l'architecture du DNS se présente sous forme d'un arbre inversé, la racine ayant la connaissance de tous les *TLD*, ayant eux-même la connaissance de l'architecture directe de leurs noeuds inférieurs. Il reste maintenant à établir comment s'effectuent les requêtes DNS, c'est-à-dire qui répond aux questions des clients et permet ainsi la résolution de noms. Deux cas se proposent à nous :

- le serveur de noms interrogé maîtrise la zone et peut donc nous renseigner,
- le serveur de noms interrogé ne maîtrise pas la zone et il faudra faire une recherche plus vaste dans l'arbre pour obtenir une réponse.

Deux méthodes ont été implémentées pour répondre à ces propositions, le mode récursif et le mode itératif.

1. Mode récursif

Dans le mode récursif, le serveur interrogé doit se charger de répondre à la demande du client, c'est à dire qu'il va interroger sa zone, puis, en l'absence de résultat, va se référer à son domaine supérieur, puis au supérieur de ce dernier, jusqu'à arriver à la racine en cas de non réponse, pour être aiguillé vers la bonne hiérarchie et obtenir une réponse satisfaisante. Le résultat de cette demande récursive peut alors être soit une réponse en bonne et due forme, soit la présentation d'une erreur, comme quoi l'hôte ou le type demandé n'existe pas. Si un des serveurs, dans le parcours de l'arbre connaît la réponse, le premier serveur chargé de la récursion peut alors demander à ce nouveau de lui indiquer quel serveur

a autorité sur le domaine. Cette nouvelle requête peut alors être faite de manière récursive ou itérative. Le mode récursif peut être considéré comme gourmand pour le serveur et peut être désactivé pour empêcher la baisse de performance du service.

2. Mode itératif

Dans le mode itératif, le serveur contacté renseigne sur les résultats qu'il obtient. À savoir, il connaît le résultat ou indique au client l'adresse d'un serveur proche, en général de niveau supérieur. C'est alors à la charge du client d'émettre une nouvelle interrogation avec les nouveaux renseignements fournis. L'escalade s'effectue ainsi jusqu'à obtenir la réponse.

3. Mémoire cache

Nous venons de voir qu'une interrogation peut entraîner un certain nombre d'échanges, ceux-ci passant souvent par la racine et les *TLD*. Afin d'alléger le processus, les serveurs configurés pour répondre en mode récursif voient transiter un grand nombre d'informations, comme les serveurs autoritaires sur un domaine donné. Ces informations sont stockées dans une mémoire cache, mémoire temporaire, qui permettra lors de nouvelles requêtes de raccourcir les recherches, en ne passant pas par les hiérarchies supérieures, mais directement en interrogeant un domaine plus proche.

Par exemple, en recherchant le serveur web du sous-domaine de l'université (`www.sciences.universite.fr.`), nous avons obtenu les domaines autoritaires pour `sciences.universite.fr.` et pour `universite.fr`. De ce fait, si nous recherchons la machine `www.lettres.universite.fr`, nous n'aurons ni besoin de remonter à la racine, ni de passer par la hiérarchie `fr` pour obtenir des informations, mais directement passer par `universite.fr`. Les réponses négatives, c'est à dire n'ayant trouvé aucun enregistrement ou données correspondant à la recherche peuvent également être stockées. Dans le premier cas, on parle de *cache positif*, dans le deuxième, de *cache négatif*.

VII Mise en place de Bind

Notre choix de distribution GNU/Linux s'étant porté sur GNU/Debian etch, il est très simple d'installer Bind. Pour cela, il faut au préalable s'assurer que nous disposons du bon dépôt de logiciels dans le fichier *sources.list* :

```
deb http://ftp.debian.org/debian/ stable main
```

Il ne nous reste plus qu'à lancer l'installation :

```
apt-get install bind
```

Il est important de noter que pour utiliser Bind avec Linux-Vservers, l'installation seule est insuffisante. il est obligatoire de recompiler l'application en activant les *capabilities*.

Après avoir téléchargé le paquet, nous retrouvons les fichiers de configuration suivants :

```
/etc/bind/db.0
/etc/bind/db.127
/etc/bind/db.255
```

```
/etc/bind/db.local
/etc/bind/db.root
/etc/bind/named.conf
/etc/bind/named.conf.local
/etc/bind/named.conf.options
```

Ces fichiers servent à la configuration du service serveur. Nous allons nous attarder sur leur utilité. Commençons par les fichiers ayant pour racine *db*. Ceux-ci définissent des espaces de nommage en relation avec les plans d'adressage IP disponibles. Ils ont ici un rôle bien précis [16] :

- Le premier fichier (db.0) contient l'adressage local qui est utilisé pour désigner le réseau sur lequel la machine se trouve : les adresses du bloc 0.0.0.0/8 font référence aux hôtes sources d'un réseau local. L'adresse 0.0.0.0/32 peut être utilisée comme adresse source pour l'hôte sur le réseau ; les autres adresses à l'intérieur de 0.0.0.0/8 peuvent être employées pour spécifier des machines sur ce même réseau.
- Le deuxième (db.127) et le quatrième fichier (db.local) font référence à la boucle locale, dite loopback : le bloc 127.0.0.0/8 est réservé à l'usage des adresses de boucle locale (loopback) des hôtes Internet. Un datagramme envoyé par un protocole de plus haut niveau sur n'importe quelle adresse de ce bloc devrait boucler au sein de l'hôte. La convention habituelle veut que l'on implémente la première adresse 127.0.0.1/32 comme loopback, mais aucune adresse incluse dans ce bloc ne devrait apparaître sur aucun réseau.
- Le troisième fichier (db.255) se rapporte au réseau de diffusion : le bloc 240.0.0.0/4, formellement connu comme l'espace d'adressage Classe E, est réservé. L'adresse de destination dite "diffusion limitée", 255.255.255.255, ne devrait pas être transmise à l'extérieur d'un sous-réseau de la source. Cet espace est limité pour un usage futur.

Ces quatre fichiers sont donc là pour empêcher une erreur de manipulation et garantir le bon fonctionnement du réseau. Il ne doivent pas être modifiés. Le fichier *db.root* contient la définition des 13 serveurs racines qui sont essentiels au fonctionnement du service DNS sur Internet. Le fichier *named.conf* contient la configuration du service serveur Bind, *named.conf.local* s'intéresse aux adresses de réseaux gérées localement. Le dernier fichier, *named.conf.options* contient les options de configuration spécifique au serveur.

1. Configuration du service serveur Bind

La configuration de Bind se déroule en deux étapes. La première consiste à déclarer les zones que notre serveur DNS gérera ainsi que les options de configuration de Bind.

```
// Include logging config file
include "/etc/bind/logging.conf";

// prime the server with knowledge of the root servers
zone "." {
        type hint;
        file "/etc/bind/db.root";
allow-transfer {none};
};

// be authoritative for the localhost forward and reverse zones, and for
```

```
// broadcast zones as per RFC 1912

zone "localhost" {
        type master;
        file "/etc/bind/db.local";
allow-transfer {none};
};

zone "127.in-addr.arpa" {
        type master;
        file "/etc/bind/db.127";
allow-transfer {none};
};

zone "0.in-addr.arpa" {
        type master;
        file "/etc/bind/db.0";
allow-transfer {none};
};

zone "255.in-addr.arpa" {
        type master;
        file "/etc/bind/db.255";
allow-transfer {none};
};

;Obtention de l'IP à partir du nom.
zone "eof.eu.org" {
        type master;
        file "/etc/bind/eof.eu.org.hosts";
        allow-transfer {86.64.61.10;};
        };

;zone reverse. Obtention du nom à partir de l'IP.
zone "61.64.86.in-addr.arpa" {
        type master;
        file "/etc/bind/db.86.64.61";
        allow-transfer {86.64.61.10;};
};
```

Sur le serveur secondaire, nous retrouvons également l'ensemble des zones nécessaires au fonctionnement du serveur DNS. Pour les zones qui nous concernent, voici la définition que nous appliquons :

```
;Obtention de l'IP à partir du nom.
zone "eof.eu.org" {
        type slave;
        file "/etc/bind/eof.eu.org.hosts";
        masters {86.64.61.10;};
```

```
        allow-transfer {none;};
        };

;zone reverse. Obtention du nom à partir de l'IP.
zone "61.64.86.in-addr.arpa" {
        type slave;
        file "/etc/bind/db.86.64.61";
        masters {86.64.61.10;};
        allow-transfer {none;};
};
```

2. Configuration des logs

La configuration des journaux de Bind peut être réglée de manière précise et puissante. Ainsi, par l'intermédiaire des *channels*, nous pouvons définir le nom du fichier (file), sa taille maximum (size), le nombre de versions conservées (version) et son niveau d'importance (severity). Lorsque la définition du *channel* est achevée, il faut alors affecter une catégorie à ce dernier. Lorsque la configuration est terminée, il suffit de redémarrer Bind pour que la journalisation commence. Voici un court exemple de configuration des logs :

```
logging {

  channel xfer-in {
file "/var/log/named/xfer-in" versions 3 size 5m;
severity dynamic;
print-time yes;
};

  channel xfer-out {
file "/var/log/named/xfer-out" versions 3 size 5m;
severity dynamic;
print-time yes;
};


  category xfer-in { xfer-in; };
  category xfer-out { xfer-out; };
```

Nous avons ici deux *channels* définis, avec le nom de leur fichier, le nombre de version que nous souhaitons conserver (ici 3) et la taille du fichier de log, à savoir 5 Mo. La sévérité est celle attribuée par défaut. Nous demandons que les logs soient précédés de la date et de l'heure de l'événement. La catégorie `xfer-in` trace les transferts de zone reçus par le serveur. La catégorie `xfer-out` trace les transferts de zone effectués par le serveur.

3. Configuration d'un domaine

a. Désignation des champs de configuration

Un certain nombre de désignations sont à notre disposition pour qualifier les différentes adresses que nous utiliserons dans notre configuration. Les enregistrements que nous trouvons dans les fichiers de zones sont appelés des *resource records* ou *RRs*. Leur constitution a une forme fixe ainsi définie [17] :

```
<name>   [<ttl>]   [<class>]   <type>   <data>
```

<name> Le nom définit quel nom de domaine est appliqué au RR courant.

<ttl> Il est important que les TTLs soient définis à une valeur bien réfléchie. En effet, ce dernier représente le temps de cache, en secondes, d'un résolveur extérieur avant qu'il ne revienne à nouveau interroger notre serveur. Si ce temps est trop court, le serveur se verra interrogé souvent, de manière inutile, puisque les données n'auront sûrement pas été modifiées. Au contraire, si ce temps est trop élevé, l'information contenue sur notre serveur ne sera pas distribuée dans un laps de temps raisonnable et les données obsolètes risquent alors de circuler sur le réseau. Si le champ est laissé vide, le TTL par défaut sera celui défini dans le SOA de la zone.

<class> Le DNS a été designé pour être indépendant du protocole. Le champ *class* permet ainsi d'identifier le groupe de protocoles dans lequel se trouve chaque RR. La classe la plus employée à l'heure actuelle, pour les utilisateurs de TCP/IP est *Internet*. La désignation standard est `IN`. Un fichier de zone devrait seulement contenir des RRs appartenant à la même classe.

<type> Cette valeur est encodée sur 16 bits, elle spécifie le type de ressource du RR. Le tableau 4.1 dresse la liste des types courants disponibles.

TAB. 4.1 – Désignation des types de champs de déclaration DNS (RRs)

Designation	Description
SOA	Start Of Authority
NS	Name Server
A	Internet Address
CNAME	Canonical Name (nickname pointer)
HINFO	Host Information
WKS	Well Known Services
MX	Mail Exchanger
PTR	Pointer

<data>

Le champ de données est défini différemment pour chaque type et classe de données. Les formats les plus courants des RR sont décrits un peu plus bas.

Nous pouvons rajouter à cette définition quelques éléments de syntaxe venant compléter la définition d'un RR. Il est à noter que le DNS n'est pas sensible à la casse. Les parenthèses ("(",")") sont utilisées pour grouper des données réparties sur plusieurs lignes. Le point-virgule (" ;") signifie le début d'un commentaire, le reste de la ligne étant ignoré. L'asterisque ("*") est utilisé comme caractère joker. Arobase ("@") fait référence au nom de domaine par défaut.

Le *SOA* désigne le début d'une zone qui prendra fin avec la déclaration du prochain SOA. Il est défini selon le schéma suivant :

```
<name>  [<ttl>]  [<class>]  SOA  <origin>  <person>  (
                          <serial>
                          <refresh>
                          <retry>
                          <expire>
                          <minimum> )
```

L'attribut `name` correspond au nom de la zone, `ttl` et `class` sont optionnels. Le champ `origin` représente l'hôte sur lequel le fichier de zone principal réside et `person` l'adresse courriel du gestionnaire de la zone. Cette dernière a la particularité de ne pas employer `@` puisque c'est un caractère spécial, mais un point en lieu et place.

<serial> est le numéro de version du fichier de zone. Celui-ci doit être incrémenté chaque fois qu'un changement de zone intervient. Par convention, on emploie souvent la date du jour suivie du numéro de modification du fichier, sous le format `YYYYMMDD##` avec `YYYY` pour l'année, `MM` pour le mois, `DD` pour le jour et `##` pour le numéro d'incrément. De ce fait, au fil des jours, on peut être certain que ce numéro sera unique. Si ce numéro n'est pas modifié, la zone sera considérée comme inchangée.

<refresh> représente le temps en secondes au bout duquel le(s) serveur(s) secondaire(s) vérifiera si le serveur primaire a connu une modification et nécessite de fait de se mettre à jour. Une bonne valeur paraît être d'une heure (soit 3600 secondes).

<retry> permet au(x) serveur(s) secondaire(s) d'essayer de nouveau l'initialisation d'une connexion avec le serveur primaire, lorsque la première tentative de *refresh* a échoué. La valeur courante est de 10 minutes (soit 600 secondes).

<expire> symbolise la limite supérieure, en secondes, dont dispose un serveur secondaire pour utiliser ses données, avant que celles-ci ne soient considérées comme obsolètes, suite à un défaut de rafraîchissement. La valeur préférée est celle de 42 jours (soit 3600000 secondes).

<minimum> est la valeur minimum d'un TTL à l'intérieur de la zone. Celui-ci est en général d'un jour (soit 86400 secondes).

```
eof.eu.org. IN      SOA     eof.eu.org. admin.eof.eu.org. (
2006120101
10800
3600
604800
38400 )
```

Le type *NS* est dédié aux serveurs de nom (*Name Server*). Il définit le serveur qui fournit le service DNS pour un domaine donné. Dans le cas d'une délégation de zone, cette définition doit se trouver à la fois dans la zone qui délègue le domaine et le domaine lui-même. Il se construit selon la nomenclature suivante :

```
<domain>   [<ttl>] [<class>]   NS   <server>
```

Une singularité est présente pour ce type de RR. En effet, si le serveur DNS se trouve lui-même dans le domaine hébergé, il est nécessaire de créer un *Glue Record*. Ce dernier est simplement l'ajout d'un enregistrement ressource de type A. Exemple :

```
eof.eu.org. IN NS ns1
ns1.eof.eu.org  IN A 86.64.61.10
```

Le type *A*, pour *Address*, associe un nom à une adresse IP.

```
<host>   [<ttl>] [<class>]   A   <address>
```

Exemple :

```
rwww IN A 86.64.61.12
```

Le *CNAME* ou *Canonical Name* est un alias d'un nom de type A. Ainsi, dans le cas d'une dématérialisation des services, on crée souvent un nom canonique, identifiant le service, en le liant à un nom d'hôte. Si le service a besoin de migrer vers un nouvel hôte, l'identification pour l'utilisateur restera le même au travers du nom canonique et pourra être modifié de manière transparente.

```
<nickname>   [<ttl>] [<class>]   CNAME   <host>
```

Exemple :

```
www1 IN CNAME rwww
```

Le type *HINFO* (Host Info) est peu employé. Il permet de définir pour chaque hôte le type de matériel ainsi que le système d'exploitation embarqué. Bien que non obsolète dans le standard DNS, la base de données de définition du matériel maintenu par l'IANA [2] n'est plus à jour depuis plusieurs années maintenant. Les informations traitées par ce type sont sensibles, seul l'administrateur du site devrait en avoir l'accès.

```
<host>   [<ttl>] [<class>]   HINFO   <hardware>   <software>
```

Exemple :

```
www HINFO DEC-2060 GNU/Linux
```

Le type *WKS* pour *Well Known Services* liste selon le référentiel de l'IANA les services à disposition, à condition que ceux-ci fonctionnent sur un port inférieur à 256 en *TCP* ou *UDP*. Tout comme *HINFO*, cet enregistrement est toujours disponible dans le standard, mais, pour des raisons de sécurité, ne devrait être adressé qu'au seul administrateur du domaine.

```
<host> [<ttl>] [<class>] WKS <address> <protocol> <services>
```

Exemple :

```
rwww WKS 86.64.61.12  TCP  SSH SMTP WWW
```

Le type *MX* employé pour *Mail Exchanger* définit l'hôte du domaine où les mails doivent être délivrés. Le champ *preference* est numérique. Il attribue une préférence d'usage des hôtes définis. Le nombre le plus petit définit le serveur favori. Le poids peut être identique pour différents serveurs. Dans ce cas, une répartition de charge se met automatiquement en place (*round-robin*).

```
<name>   [<ttl>] [<class>]   MX   <preference>   <host>
```

Exemple :

[2] Internet Assigned Numbers Authority http ://www.iana.org

```
alpha MX 10 www1
beta MX 20 www2
delta MX 30 www3
```

Le dernier type, *PTR*, est différent des autres approches puisqu'il définit l'enregistrement dans le sens inverse, c'est à dire qu'à partir de l'adresse IP, il permet de retrouver le nom associé. Pour ce faire, le domaine `IN-ADDR.ARPA` a été créé, afin de permettre une indexation aisée, impossible sans ce domaine commun. Pour respecter l'ordre arborescent du schéma DNS, l'adresse IP présente dans le RR doit être inversée en commençant par le dernier digit, pour remonter jusqu'à la racine de l'arbre. Un enregistrement *PTR* devrait toujours pointer vers un enregistrement de type `A` et non de type `CNAME`.

```
<special-name>   [<ttl>] [<class>]   PTR   <name>
```

Exemple :

```
12.61.64.86.IN-ADDR.ARPA. IN PTR rwww
```

b. Mise en place des fichiers de zone

Pour les besoins d'implémentation de la maquette, nous devons déclarer deux zones. Une première à destination d'Internet, qui permettra de joindre les serveurs depuis l'extérieur, une deuxième, interne, qui référencera l'intégralité des serveurs avec une affectation dans un réseau privé. Voici donc le fichier de déclaration de zone externe :

```
$ttl 38400 ;durée de vie du cache

@ IN      SOA     eof.eu.org. admin.eof.eu.org. (
2006120101 ; numéro de série
10800 ; refresh
3600 ; retry
604800 ; expire
38400 ) ; minimum

@ IN      NS      ns1 ; Serveur dns
@ IN      NS      ns2 ;

ns1 IN      A       86.64.61.10 ; GLUE record
ns2 IN      A       86.64.61.11 ;
www1 IN      CNAME rwww ; Alias des serveurs web
www2 IN      CNAME rwww ;
www3 IN      CNAME rwww ;
rwww IN A 86.64.61.12 ; Répartiteur de charge
hssh IN A 86.64.61.13 ; Security pleasure
vssh IN A 86.64.61.15 ; Vserver
```

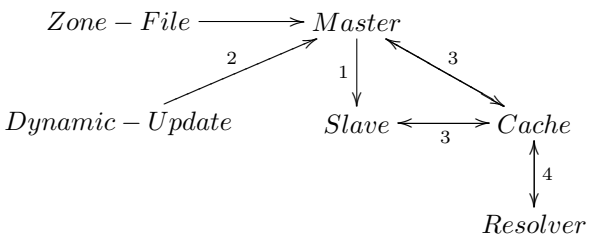
Dans ce premier fichier, on constate que seuls les serveurs DNS, le répartiteur de charge qui vient en frontal des serveurs web et les serveurs ssh (honeypot et serveur d'administration) sont présents. Ainsi, les autres serveurs, qui n'ont que des adresses privées, n'ont pas besoin d'être renseignés, ni d'être connus de l'extérieur.

```
$ttl 38400
@ IN      SOA     eof.eu.org. admin.eof.eu.org. (
2006120101 ; numéro de série
10800 ; refresh
3600 ; retry
604800 ; expire
38400 ) ; minimum

@ IN      NS      ns1 ; Serveur dns
@ IN      NS      ns2 ;

ns1 IN      A       192.168.0.1 ; GLUE record
ns2 IN      A       192.168.0.2 ;
fw IN A 192.168.0.3
www1 IN      A 192.168.0.4 ; Alias des serveurs web
www2 IN      A 192.168.0.5 ;
www3 IN      A 192.168.0.6 ;
rwww IN A 192.168.0.7 ; Répartiteur de charge
hssh IN A 192.168.0.8 ; Security pleasure
vssh IN A 192.168.0.9 ; Vserver
ids IN A 192.168.0.10 ; Détection d'intrusion
syslog IN A 192.168.0.11 ; Archivage des logs
pki IN A 192.168.0.12 ; Authentification
```

VIII Sécurité

Nous l'avons vu dans la présentation précédente, le *DNS* ainsi que les informations qu'il délivre et qu'il stocke sont primordiaux pour le fonctionnement du réseau. De ce fait, une attaque contre un serveur dns peut mettre en péril l'ensemble de l'architecture. À ce défaut de conception viennent se greffer deux couches de sécurité et une optimisation à la configuration. Ainsi, les listes de contrôles d'accès permettent à l'administrateur de gérer les différents espaces qu'il adresse par groupe, alors que la *Transaction Signature* ou le *DNSSEC* viendront renforcer notre chaîne.

La figure 4.2 relate les différentes attaques connues sur le protocole *DNS* :



TAB. 4.2 – Flux DNS et vulnérabilités

Pour accompagner ce schéma, voici le descriptif des attaques ainsi que les solutions proposées pour les rendre nulles et non-avenues :

1. falsification d'identité du master, corruption du transfert de zone, (TSIG)

2. mises à jour non autorisées, contenu corrompu, (TSIG)
3. corruption du trafic DNS (DNSSEC)
4. falsification d'identité du cache, corruption du trafic DNS (TSIG)

Nous avons maintenant identifié les outils qui vont nous permettre d'enrayer les menaces, regardons maintenant comment les mettre en place au sein de notre architecture.

1. ACL

Les listes de contrôle d'accès (*acl*) permettent de définir de manière intelligente un ensemble d'adresses IP ou de réseaux et de réutiliser ces acl dans les sous-structures, comme *allow-query* ou *allow-transfer*. La syntaxe de la déclaration d'acl est la suivante :

```
acl acl-name {
    address_match_list
};
```

Quatre directives sont prédéfinies pour répondre aux besoins courants. Il est donc possible de les appeler directement :

any : Autorise tous les hôtes.

none : Interdit tous les hôtes.

localhost : Autorise les adresses IP de toutes les interfaces du système.

localnets : Autorise n'importe quel hôte d'un réseau pour lequel le serveur a une interface.

Voici un exemple d'implémentation des acl dans le fichier de configuration de Bind, qui, pour notre architecture, permet d'interdire l'accès aux réseaux appartenant à `non_acces`, et au contraire autoriser l'accès et l'interrogation au réseau de confiance de notre infrastructure noté `interne` :

```
; Adresses répondant à la rfc 1918
acl non_acces {
0.0.0.0/8;
1.0.0.0/8;
2.0.0.0/8;
224.0.0.0/3;
127.0.0.0/8;
169.254.0.0/16;
10.0.0.0/8;
172.16.0.0/12;
};

; Défnition du réseau
acl interne {
192.168.0.0/24;
};

; Définition des options
options {
```

```
  allow-query { interne; };
  allow-recursion { interne; };
;liste des adresses auxquelles notre serveur ne répondra pas
  blackhole { non_acces; };
};
```

2. TSIG

La *Transaction SIGnature* est un protocole [18] qui permet l'échange de données au travers du partage d'une clé crytpographique. Sans cette dernière, toute tentative de transfert de zone, de mise à jour dynamique ou de perturbation de transaction (corruption de trafic, falsification de cache) devient impossible. La clé est symétrique, ce qui veut dire que le chiffrement et le déchiffrement s'effectue par l'intermédiaire de cette seule et même clé. Lors du transfert, ce ne sont pas les données qui sont chiffrées (contrairement à ce que nous verrons dans *DNSSEC*), mais l'empreinte du fichier de zone à transférer. Cela permet un échange beaucoup plus rapide (moins de données), ainsi qu'un emploi limité du processeur pour le chiffrement des zones. Passons à la mise en place sur notre serveur. Il nous faut au préalable créer la clé partagée qui nous servira pour l'échange des zones. Pour ce faire, on emploie la commande suivante :

```
dnssec-keygen -r /dev/urandom -a HMAC-MD5 -b 128 -n HOST ps.tsig
```

qui va générer une paire de clés.

```
Kps.tsig.+157+01300.key  Kps.tsig.+157+01300.private
```

Les deux clés contiennent chacune la chaîne de caractères qui nous permettra d'effectuer l'échange. Il suffit d'en extirper son contenu :

```
cat Kps.tsig.+157+01300.key
ps.tsig. IN KEY 512 3 157 0YjtpKUAT0TLPHXVd4mTaA=
```

et de le placer dans un fichier de description de la clé compréhensible par Bind :

```
cat /etc/bind/shared.keys
key ps.tsig. {
        algorithm hmac-md5;
        secret "0YjtpKUAT0TLPHXVd4mTaA==";
};
```

Le plus difficile est à présent effectué. Il reste à indiquer au serveur d'utiliser la clé :

```
include "/etc/bind/shared.keys";
server 192.168.0.2 {
   keys { ps.tsig; };
};
```

et modifier la zone sur le maître, en changeant l'autorisation de transfert. On ne fait plus pointer vers le serveur secondaire, mais on applique la clé cryptographique :

```
zone "eof.eu.org" {
        type master;
        file "/etc/bind/eof.eu.org.hosts.signed";
        allow-transfer { key ps.tsig.;};
        };
```

Par la suite, la clé doit également être distribuée sur le ou les serveurs secondaires, en indiquant l'adresse du serveur primaire pour le partage de la clé :

```
include "/etc/bind/shared.keys";
server 192.168.0.1 {
   keys { ps.tsig; };
};
```

Aucune modification sur le descriptif de la zone n'est nécessaire. Notre solution de *Transaction Signature* est fonctionnelle.

3. DNSSEC

a. présentation

Le protocole DNS est simple dans sa conception. Cette simplicité lui vaut de pouvoir être facilement détourné à différents endroits. Nous avons vu comment *TSIG* cloisonne sur plusieurs fronts les transferts de zone, il nous est maintenant nécessaire de nous attarder un peu plus sur la mise en place d'une signature sur les enregistrements DNS. En effet, par défaut, aucun élément ne permet d'affirmer que les données reçues par l'intermédiaire des réponses n'ont pas été modifiées ou altérées. Pour ce faire, nous allons donc signer numériquement, c'est à dire créer une empreinte, de chaque enregistrement sur le serveur DNS. Cette empreinte est placée dans le fichier de zone, référencée sous le type *SIG*. Cela n'est cependant pas suffisant pour garantir l'intégrité des données reçues par le client. On place donc également la clé publique, pendante de la clé privée qui sert à signer les enregistrements, dans le fichier de zone. Par conséquent, lors de la réception de la réponse DNS, le client sera en mesure de vérifier l'authenticité de l'enregistrement fourni par la réponse en le confrontant avec la clé publique (voir le chapitre sur l'infrastructure de clé publique pour plus de détail sur le mécanisme cryptographique). Ce système de signature a d'autres avantages que celui d'assurer l'intégrité des données transmises. Par exemple, en cas de compromission du serveur primaire, si la clé privée n'est pas déposée sur le dit serveur, le pirate ne pourra en aucun cas modifier les enregistrements. Un autre avantage, voire une flexibilité, est offert. Il est envisageable de créer une zone et la signer sur une autre machine que le serveur, *machine de confiance*, avant de l'importer ce qui écartera la possibilité d'employer un binaire corrompu pour la génération des zones.

Il reste la problématique de la recherche dans l'arbre DNS. En l'état, notre zone parent contient un certain nombre d'informations à propos de notre domaine, à commencer par nos serveurs de noms. Si ceux-ci sont modifiés à notre insu, la sécurisation de notre serveur sera inutile, puisqu'il ne sera plus consulté. Pour parer à ce problème, le *Delegation Signer* ou *DS* se compose de l'identifiant et de l'empreinte de la clé de notre zone. Ce DS est déposé sur le serveur parent et signé par sa propre clé privée. Lors d'une requête, le client passera donc par notre parent qui pourra lui indiquer de manière *certaine* que nos serveurs sont bien les serveurs autoritaires sur la zone recherchée.

b. Mise en pratique

Passons maintenant à la mise en place de DNSSEC [19][20] dans notre architecture. Dans notre étude, ce procédé n'est employé que pour la vue privée, c'est à dire une utilisation locale des ressources, afin de s'assurer que les serveurs communiquent bien avec qui ils croient le faire.

Pour activer le mécanisme *DNSSEC* sur notre serveur Bind, il est nécessaire de le préciser dans les options, sans quoi les réponses retournées ne contiendront pas la signature souhaitée :

```
options {
// DNSSEC should be turned on, do not forget
dnssec-enable yes;
}
```

L'étape suivante consiste à la création de deux paires de clés. Comme nous l'avons évoqué précedemment, un DS est mis en place sur notre serveur parent. Cependant, si nous souhaitons modifier de manière répétée nos clés de chiffrement, cela ne sera possible sans faire une modification du DS répertorié sur le serveur parent. Pour résoudre ce problème, nous créons donc une première paire de clés nommée *key-signing key* (KSK) qui sera la clé de référence pour le DS, avec une grande taille (4096 bits par exemple), et qui permettra également de signer numériquement la deuxième paire de clés, nommée *zone-signing key* (ZSK), qui, d'une taille inférieure, aura la charge de signer directement les enregistrements des zones.

Nous obtenons ainsi la chaîne de confiance désirée : nous pouvons avoir confiance dans le serveur hiérarchique, qui fournit les adresses des NS de notre zone, ainsi que le DS associé. Ce DS permet la vérification de la *KSK* présente sur notre serveur autoritaire. Cette même KSK valide notre *ZSK* qui authentifie nos données. Il est maintenant possible de changer aussi souvent que souhaité la ZSK sans pour autant devoir faire modifier le serveur hiérarchique.

Nous passons donc à la création du couple de clés *KSK* :

```
dnssec-keygen -r/dev/urandom  -f KSK -a RSASHA1 -b 1280  -n ZONE eof.eu.org
```

qui génère les deux fichiers suivants :

```
Keof.eu.org.+005+04963.key  Keof.eu.org.+005+04963.private
```

Le contenu de la clé publique montre que le drapeau *ksk* a ajouté un bit, ici la valeur est passée de 256 à 257 :

```
eof.eu.org. IN DNSKEY 257 3 5 AwEAAd[...]vajic=
```

À titre indicatif, voici également le contenu du fichier contenant la clé privée :

```
Private-key-format: v1.2
Algorithm: 5 (RSASHA1)
Modulus: 1bVBJQ1PV[...]q9qOJw==
PublicExponent: AQAB
PrivateExponent: Uo8ROaUorq3jFHv1bY[...]Now2/tEQ==
Prime1: +3nokVVrQ2YPxZr4kDR0rGh99Qt[...]x6pfjIRm0=
Prime2: 2Y1qv6ynuYxL0zb3hkPeKX7KNbO[...]9BJMEZWmM=
Exponent1: UMaw0HYKlUSkTkngC5n93B9Y[...]rEHqnR0uU=
Exponent2: XV5bRT41keXLmS/LFf3EuGxp[...]zaCYsydV8=
Coefficient: 64aVDbHOQZyg2CZt/tS9Pa[...]rbGC9MYE8=
```

Nous procédons de la même manière pour créer le couple de clé *ZSK*, avec une taille un peu moins grande :

```
dnssec-keygen -r/dev/urandom  -a RSASHA1 -b 1024 -n ZONE eof.eu.org
```

Nous obtenons les fichiers suivants :

```
Keof.eu.org.+005+55245.key  Keof.eu.org.+005+55245.private
```

Pour terminer le processus, nous signons la zone concernée à l'aide de la *KSK* et *ZSK*. Pour ce faire, nous ajoutons les enregistrements à la fin du fichier de zone :

```
$include Keof.eu.org.+005+04963.key
$include Keof.eu.org.+005+55245.key
```

puis nous enclenchons la signature de la zone :

```
dnssec-signzone -o eof.eu.org.hosts -k Keof.eu.org.+005+04963.key\
/etc/bind/eof.eu.org.hosts Keof.eu.org.+005+55245.key
```

qui va générer le nouveau fichier `eof.eu.org.hosts.signed` avec les enregistrements *SIG*.

Il faut penser à modifier le nom du fichier de description de zone dans le fichier de configuration de bind :

```
zone "eof.eu.org" {
        type master;
        file "/etc/bind/eof.eu.org.hosts.signed";
        allow-transfer {86.64.61.10;};
        };
```

Une fois le serveur correctement configuré, on peut vérifier par l'intermédiaire de la commande *dig* que le *dnssec* est bien en place.

```
; <<>> DiG 9.3.4 <<>> +dnssec +multiline -t key eof.eu.org @192.168.0.2
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1212
;; flags: qr aa rd; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;eof.eu.org.            IN KEY

;; AUTHORITY SECTION:
eof.eu.org.             38400 IN SOA eof.eu.org. admin.eof.eu.org. (
                                2007022101 ; serial
                                10800      ; refresh (3 hours)
```

```
                                3600       ; retry (1 hour)
                                604800     ; expire (1 week)
                                38400      ; minimum (10 hours 40 minutes)
                                )
eof.eu.org.             38400 IN RRSIG SOA 5 3 38400 20070323212054 (
                                20070221212054 37725 eof.eu.org.
                                ODbmzcEKCEZFdRq+DoAwlZfQC50Rp2sp1DQSoLeFGkBP
                                YSPAwi0rrf8VNCMoaSQYpdmOkaDoNrh0S5WABZ+rXEt0
                                pBJBijrg2iinnpBRPPnshSGCwBndF6Jk7ezJWZNq )
eof.eu.org.             38400 IN NSEC fw.eof.eu.org. NS SOA RRSIG NSEC DNSKEY
eof.eu.org.             38400 IN RRSIG NSEC 5 3 38400 20070323212054 (
                                20070221212054 37725 eof.eu.org.
                                HTrAudzoZIwTUh02WntG1ETOKBtCr8P1titKMa/69cmx
                                Ge7CyZEaIv5zjwGvWeGVKVp8mdB2/jUMMuGlHo+J475B
                                Vk87zNTcgM91XoUyfx5DiidkauTRztjy5Vq+F7c2 )

;; Query time: 71 msec
;; SERVER: 192.168.0.2#53(192.168.0.2)
;; WHEN: Thu Feb 22 22:41:22 2007
;; MSG SIZE  rcvd: 402
```

De même, si on interroge un hôte, la signature est bien présente.

```
; <<>> DiG 9.3.4 <<>> +dnssec +multiline -t a fw.eof.eu.org @192.168.0.2
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18987
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;fw.eof.eu.org.          IN A

;; ANSWER SECTION:
fw.eof.eu.org.          38400 IN A 192.168.0.3
fw.eof.eu.org.          38400 IN RRSIG A 5 4 38400 20070323212054 (
                                20070221212054 37725 eof.eu.org.
                                RzYFrXjj0a8b7RPy5IXQngQ6V8237G4Ukb5hnbebF62g
                                /Uv4LtriuNYUbZMvoWlRZriKgHKxkQAlTplTf4T8i4eT
                                Fr2/MR6oY1bXcCtZr3qF9++lSr3ytuOdHSHvrDka )
```

Nos serveurs primaire et secondaire, leurs échanges, les mises à jour ainsi que les interrogations sont maintenant bordés et sécurisés. Notre système DNS est fonctionnel, nous pouvons attaquer la mise en place des différents éléments de notre infrastructure.

Chapitre 5

Le Web

I Histoire

Si le réseau a connu sa première flamme avec ARPAnet, commandé par le *Department Of Defense*, c'est bien avec l'arrivée de *TCP/IP*, inventé par Vinton Cerf, que le *web*, tel que nous le connaissons, a commencé à exister. Effectivement, quelques protocoles sont déjà présents, comme le courrier électronique ou encore les *newsgroups*, mais c'est bien le *World Wide Web* qui popularisera, au cours des années, l'usage que nous en avons.

Tim Berners-Lee, qui est à l'époque employé au *Conseil Européen pour la Recherche Nucléaire* (*CERN*) construit un modèle qui permet l'édition, la consultation et où chaque page est liée à une autre. C'est l'ancêtre du *wiki* qui porte alors le nom de projet *ENQUIRE*. Ce premier projet lui permettra de se frotter au modèle des *hyperliens*. Quelques années plus tard, en 1989, il publie un papier sur *une large base de données de liens hypertext basée sur des liens classés* [1].

N'ayant aucun succès auprès des industriels, Berners-Lee et son collègue et contributeur au projet, Robert Cailliau, installent le *World Wide Web* au CERN et obtiennent l'autorisation de porter l'annuaire téléphonique de l'institut sur ce média, afin de l'aider à se populariser. Berners-Lee développe un navigateur nommé simplement WorldWideWeb qui permet, au-delà de la navigation sur le web, d'accéder aux newsgroups et aux serveurs FTP. Ce navigateur mono plate-forme sera repris par Nicola Pello qui écrira WWW, un navigateur en ligne de commande disponible sur la plupart des plates-formes de l'époque.

C'est avec la visite de Paul Kunz que le net que nous connaissons prendra de l'ampleur. Il travaille au *Stanford Linear Accelerator Center*. Il est tout de suite séduit par le principe exposé par Berners-Lee et ramènera dans ses valises de quoi créer un site vitrine pour son organisme. C'est le premier site en dehors du CERN, mais également le premier aux États-Unis d'Amérique.

Le *National Center for Supercomputing Applications* (NCSA) de l'Université de l'Illinois continuera le développement d'un navigateur par l'intermédiaire de deux de ses étudiants. Il en résultera *Mosaic*, qui est à la base d'une bonne partie des navigateurs modernes (netscape, mozilla, MSIE, etc). C'est également au sein du NCSA que le serveur web le plus populaire verra le jour. Le *HTTP daemon* ou *httpd* servira de base au développement d'*Apache*, le serveur web le plus en vogue actuellement (plus de 60% de part de marché [2]).

[1] http ://www.w3.org/Administration/HTandCERN.txt
[2] http ://news.netcraft.com/archives/web_server_survey.html

Le web s'appuie sur des standards ouverts tels que *TCP/IP* ou *DNS*. Il apporte également son lot de standards, avec entre autres *HTTP*, *HTML*, *CSS* ou encore *XML*.

II Syntaxe d'une URL

Une *URL* [21] (*Uniform Resource Locator*) est une chaîne compacte représentant une ressource disponible sur Internet. Ce concept est dérivé des *URI* [22] (*Uniform Resource Identifier*) présentes dans les spécifications du WWW développé par Berners-Lee. Cette chaîne peut se composer de l'ensemble des caractères alphanumériques augmentés des caractères « + », « . » et « − ». La casse n'a aucun impact sur l'hôte, par contre, elle est prise en compte pour le chemin de la ressource. Cependant, il se peut que, pour une raison quelconque et exceptionnelle, un caractère soit appelé et que celui-ci n'appartienne ni à cet ensemble de caractères, ni à la table de référence *US-ASCII*. Dans ce cas, il faudra encoder ce caractère à l'aide d'un trigramme composé du signe pourcentage (%) et de deux caractères alphanumériques en majuscule. Par exemple, la représentation du caractère espace sera faite par l'intermédiaire du trigramme « %20 ». le tableau 5.1 donne la liste des caractères à risque. Ils sont nommés ainsi car ils peuvent être employés par les différents langages ou protocoles utilisant les URLs. De fait, ils doivent obligatoirement être encodés lors de leur usage. Bien qu'il soit possible d'encoder un caractère réservé, il faut cependant être prudent car cela pourrait changer l'interprétation de l'URL.

TAB. 5.1 – Liste des caractères à risque, réservés et spéciaux pour une URL

Caractères à risque	« > », « < »,« # », « % », « " », « { », « } », « | », « \ », « ^ », « ~ », « [», «] », « ` ».
Caractères réservés	« ; », « / », « ? », « : », « @ », « = », « & ».
Caractères spéciaux	« $ », « − », « _ », « . », « + », « ! », « * », « ' », « (», «) ».

Le schéma URL HTTP est utilisé pour désigner une ressource accessible par l'intermédiaire du protocole *HTTP* [23] (*HyperText Transfer Protocol*). Il prend la forme suivante :

```
http://<host>:<port>/<path>?<searchpart>
```

La variable `host` est définie par un nom de domaine pleinement qualifié ou un nom d'hôte, ou encore par une adresse *IP*. Dans le cas d'un nom de domaine pleinement qualifié, les libellés de domaine commencent et finissent par un caractère alphanumérique et peuvent contenir le caractère « - ». La plupart des domaines ne commenceront pas par un chiffre, puisque c'est syntaxiquement ce qui différencie un nom de domaine d'une adresse *IP*.

Le `port` désigne le numéro de port auquel on souhaite se connecter. Par défaut, pour *HTTP* et *HTTPS*, ce sont les ports 80 et 443. Il est possible de forcer la négociation sur un autre port en indiquant le nouveau numéro de port, séparé par le caractère `:`.

Contrairement à d'autres protocoles, la norme n'autorise pas l'utilisation d'un compte et d'un mot de passe dans l'URL. Cependant, l'implémentation de l'authentification basique a pour résultat sa présence.

Le `path` (chemin) est un sélecteur HTTP et `searchpart` représente une chaîne de recherche. Le chemin est optionnel, tout comme la chaîne de recherche (précédée par le caractère `?`). Il se peut que `path` et `searchpart` ne soient pas présents. Dans ce cas, le caractère final `/` n'est pas obligatoire.

III Dialogue entre le client et le serveur

Le protocole HTTP se base sur une relation client/serveur. Nous allons définir chacun de ces termes afin de comprendre le rôle de ceux-ci. Un client est un programme qui établit des connexions dans le but d'envoyer des requêtes. Un serveur, quant à lui, est un programme qui reçoit les requêtes et possède les fonctionnalités suffisantes pour y répondre. Le schéma de dialogue du point de vue du serveur est le suivant :

- étape 1 : le serveur accepte les connexions clientes,
- étape 2 : il réceptionne des requêtes,
- étape 3 : traite les requêtes,
- étape 4 : met en place la liaison et l'accès aux ressources demandées,
- étape 5 : construit la réponse,
- étape 6 : envoie la réponse,
- étape 7 : archive la requête.

La réponse peut être un succès ou un échec. Dans le cas de l'échec, ce dernier est considéré comme une réponse à la requête et est également archivée.

Lorsque ces sept étapes sont validées, la transaction HTTP est considérée comme complète et valide. L'étape 7, bien que positionnée en dernier, est importante pour l'administrateur, puisque c'est la seule, en dehors des données du pare-feu, qui permet de retracer l'historique de la vie du serveur web, tant les interrogations réussies que les échecs, qui peuvent servir à la fois pour détecter des erreurs de configuration, de comportement, mais également des tentatives d'intrusion.

IV Les différentes méthodes

Les étapes 2 et 3 présentées dans la section précédente s'orientent sur les requêtes formulées par le client. Celles-ci se composent d'ordres, également appelées méthodes qui permettent un dialogue simple mais efficace avec le robot du serveur web. Voici la liste des ordres ainsi que leur but :

- GET : permet de récupérer le document spécifié par l'URL.
- POST : permet d'envoyer des informations au serveur.
- HEAD : permet de ne récupérer que la partie en-tête d'une réponse complète.
- OPTIONS : permet au client de connaître les options de communication utilisables pour obtenir la ressource.
- PUT : permet d'envoyer au serveur un document à enregistrer à l'URI spécifiée.
- DELETE : efface la ressource spécifiée.
- TRACE : méthode de contrôle. Demande au serveur de renvoyer la requête telle qu'elle a été reçue.
- CONNECT : cette méthode est réservée pour les proxy permettant de faire du tunneling.

Ainsi, on peut très bien récupérer l'entête d'une page Internet à travers la ligne de commande en utilisant l'instruction HEAD. Nous utilisons pour cela le logiciel *netcat* (commande `nc`) qui permet de lire et d'écrire des commandes à travers un flux réseau :

```
jop@Faith:~ nc ferry.eof.eu.org 80
HEAD / HTTP/1.0
```

qui retourne les informations suivantes :

```
HTTP/1.1 200 OK
Date: Thu, 26 Jul 2007 22:06:51 GMT
Server: Apache
X-Powered-By: PHP/5.0.5-3
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: SessionId=20021ac563581160d08fe4be69db3ed2
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

À savoir, entre autre que la page a bien été trouvée (code retour positionné à 200), la date de l'interrogation, la version de PHP utilisée, ainsi que le type de la page (HTML) et la langue du serveur (ISO-8859-1, c'est à dire encodage latin1).

V Les codes de statut

Les codes de statut, également appelés code retour dans l'exemple précédent, indiquent l'état de la réponse retournée par le serveur. Ils sont définis dans la liste ci-dessous :

- 1xx : Information - requête reçue, le processus continue,
- 2xx : Succès - l'action demandée a été reçue avec succès, comprise et acceptée,
- 3xx : Redirection - une action complémentaire doit être prise pour compléter la requête,
- 4xx : Erreur côté client - la requête contient une mauvaise syntaxe ou ne peut être satisfaite,
- 5xx : Erreur côté serveur - le serveur est dans l'incapacité de répondre à une requête apparemment valide.

Ainsi, l'erreur commune `404` relate l'interrogation d'une page non disponible sur le serveur, le `403` indique la tentative de consultation d'une ressource non autorisée, alors que le 500 fait part d'une erreur du serveur hébergeant le site. Ces codes retours sont donc importants pour la compréhension du dialogue entre le client et le serveur, mais également pour l'utilisateur final. L'administrateur doit veiller à personnaliser les codes d'erreur afin que ce ne soient pas des messages difficilement compréhensibles qui apparaissent à l'utilisateur, mais bien une page en correspondance avec la charte graphique et usuelle du site.

Vous pourrez retrouver en annexe D l'intégralité des codes retours en usage à ce jour.

VI La sécurité en question

Le SANS Institute [3], organisme le plus connu dans le domaine de l'éducation, de la recherche et de la certification en matière de sécurité, classe les attaques sur les applications web comme le risque majeur de piratage actuel. De nombreuses applications telles que les outils de Gestion de Contenu (CMS), les wikis, les portails, les forums sont autant d'applications mises à la disposition des utilisateurs, mais

[3] http://www.sans.org

aussi des malveillants. Ainsi, il ne se passe pas une semaine sans que des centaines de vulnérabilités ne soient rapportées. Il en résulte des centaines de milliers, voire des millions d'attaques vers ce type de logiciel. Il est dès lors important de prendre en compte cette menace le plus tôt possible dans notre chaîne de publication, c'est à dire au moment de la conception des applications. Le protocole HTTP n'a pas été créé pour gérer la sécurité, seule la question de l'authentification et de la confidentialité est abordée.

1. L'authentification

L'authentification permet de reconnaître et autoriser les utilisateurs.

a. Les cookies

Le cookie est un enregistrement de petite taille qui sert de référence à un site internet. Il permet à la fois de sauvegarder les informations relatives à l'utilisateur.

```
.amazon.fr      TRUE    /       FALSE   ubid-acbfr      403-2986433-9490128
.amazon.fr      TRUE    /       FALSE   session-id-time 1186869600l
.amazon.fr      TRUE    /       FALSE   session-id      402-9461430-7425715
```

Ces informations ne sont pas forcément compréhensibles, pour une lecture humaine, comme nous pouvons le voir dans l'exemple présenté ci-dessus. Elles ne sont exploitables que pour le site qui a déposé ce *cookie* à travers le navigateur de l'utilisateur. Certaines applications usent de ce procédé pour conserver le mot de passe. Il est conseillé, cependant, de ne pas stocker de mot de passe, ou de niveau d'authentification dans le cookie, celui-ci pouvant facilement être modifié par l'utilisateur.

De nombreux langages, tels que *PHP*, permettent également l'utilisation des cookies côté serveur, appelés *cookie de session*. Ceux-ci sont les pendants des cookies utilisateurs, mais sont stockés sur le serveur.

b. Basic Authentication

L'authentification basique permet d'associer un utilisateur et son mot de passe pour lui permettre l'accès à un répertoire restreint. Pour cela, apache propose de configurer un type d'authentification, ici `basic` en opposition à `digest` encore en expérimentation, suivi du nom que porte la bannière, le chemin d'accès au fichier de mots de passe, ainsi que la directive qui autorise toute connexion lorsque l'utilisateur est correctement authentifié :

```
AuthType Basic
AuthName "Accès restreint"
AuthUserFile /usr/local/apache2/.password
Require valid-user
```

Dès lors que la configuration est en place, il faut procéder à la création du fichier qui contient les comptes, ici `.htpasswd` :

```
htpasswd -c -m .passwd jpgaulier
New password:
Re-type new password:
Adding password for user jpgaulier
```

Voici le résultat du fichier créé :

```
jpgaulier:$apr1$bv1XL/..1i12gTHX2YFWas4JAlaT7a/
```

2. Le chiffrement : HTTPS

Comme nous l'exposions plus tôt dans ce chapitre, le protocole HTTP fait transiter en clair les données des utilisateurs. Dès lors que l'on souhaite que ces données soient confidentielles, c'est à dire que seuls le client émetteur et le serveur destinataire puissent connaître le contenu de celles-ci, il est nécessaire de mettre en place une couche cryptographique qui chiffrera les données pendant le transit.

Dans le chapitre sur l'Infrastructure de Gestion de Clé, le logiciel déployé permet la création et la gestion des certificats, nous ne présenterons ici que les directives à renseigner pour que le serveur prenne en compte la couche SSL :

```
Listen 443
SSLEngine on
SSLCertificateFile /usr/local/apache2/conf/ssl.crt/server.crt
```

La directive `Listen` configure l'écoute sur le port 443. On active le moteur SSL avec l'option `SSLEngine`. Pour finir, on configure le chemin d'accès au certificat du serveur.

Il est important de noter qu'une adresse IP peut porter un et un seul certificat. Ainsi, il n'est pas possible d'héberger plusieurs sites avec des noms de domaine différents sur une même adresse IP avec un certificat dédié pour chacun. Seul le premier certificat sera pris en compte. Pour les autres noms de domaine, un message d'alerte signifiera un problème d'incompatibilité.

VII Les hôtes virtuels

Un serveur web transpose le contenu d'un répertoire vers une information consultable sans autre outil qu'un navigateur, à travers le protocole HTTP et un serveur. Celui-ci écoute par design sur le port 80. Cependant, mettre en place une ressource pour chaque site s'avère gourmand en ressources et peu évident. Pour répondre à la gestion multi-site à partir d'un seul serveur, les développeurs ont créé le concept d'*hôtes virtuels*. Chaque hôte se voit attribuer ses propres caractéristiques. Dans l'exemple suivant, nous présentons deux sites. Le premier est le site institutionnel, qui fonctionne sur le port 80, dont le nom principal du serveur est `eof.eu.org` et l'alias préfixé de `www`. On indique également le chemin dans lequel sont répertoriées les données du site, ainsi que le nom du fichier d'erreur que l'on souhaite. Pour le deuxième site, on retrouve la même syntaxe, seuls les paramètres changent.

```
<VirtualHost *:80>
 ServerAdmin admin@eof.eu.org
 ServerName eof.eu.org
 ServerAlias www.eof.eu.org
 DocumentRoot /var/www/eof/
 ErrorLog /var/log/apache2/eof/error_log
</VirtualHost>

<VirtualHost *:80>
```

```
 ServerAdmin admin@eof.eu.org
 ServerName ferry.eof.eu.org
 DocumentRoot /var/www/ferry/
 ErrorLog /var/log/apache2/ferry/error_log
</VirtualHost>
```

Il est important de noter que si les répertoires qui contiennent les fichiers de journalisation n'existent pas, le serveur ne démarrera pas.

VIII Mise en place du serveur

1. Installation du serveur

Comme lors de chaque installation, nous utilisons le gestionnaire de paquet interne à notre distribution, Debian GNU/Linux, pour obtenir la dernière version packagée du serveur web. Nous utiliserons Apache [4] version 2, qui est le serveur le plus connu et le plus utilisé de par le monde :

```
jop@ema:~ sudo aptitude install apache2
Reading package lists... Done
Building dependency tree... Done
Reading extended state information
Initializing package states... Done
Reading task descriptions... Done
Building tag database... Done
The following NEW packages will be automatically installed:
  apache2-mpm-worker apache2.2-common
The following packages have been kept back:
  libapache-mod-php4 libfreetype6 libpq4 php4 php4-common php4-gd
  php4-mysql php4-pgsql
The following NEW packages will be installed:
  apache2 apache2-mpm-worker apache2.2-common
0 packages upgraded, 3 newly installed, 0 to remove and 8 not upgraded.
Need to get 1391kB of archives. After unpacking 4284kB will be used.
Do you want to continue? [Y/n/?]
```

Quelques paquets complémentaires sont mis en place. L'installation est à présent terminée, il ne reste plus qu'à configurer le serveur pour délivrer notre service.

2. Configuration du serveur

Nous n'aborderons pas la configuration complète d'un serveur apache ainsi que l'optimisation qui peut en découler ; c'est une opération longue et fastidieuse qui pourrait prendre un ouvrage à elle toute seule. Nous évoquerons ici les points importants.

Les deux premiers paramètres que nous positionnons permettent de réduire le *bruit* émis par le serveur. Nous allons tout d'abord lui demander de ne pas afficher ses numéros de version lorsqu'il communique avec un client, pour cela nous désactivons le mode signature :

[4] http://httpd.apache.org

```
ServerSignature off
```

Ensuite, nous passons du mode verbeux au mode production, ce qui signifie que toutes les versions de modules (php, ssl...) ne seront désormais plus accessibles via les entêtes du serveur :

```
ServerTokens Prod
```

Le paragraphe suivant s'occupe de la gestion du serveur lui-même, avec le chemin du fichier de verrou qui indique que le serveur est en fonctionnement, le fichier qui conserve le numéro de processus du serveur. Le `Timeout` s'intéresse au temps maximum entre le début d'une requête et la fin de cette dernière, alors que la notion de `KeepAlive` dépend du nombre de requêtes autorisées à passer sur une même connexion TCP.

```
LockFile /var/lock/apache2/accept.lock
PidFile /var/run/apache2.pid
Timeout 300
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 15
```

Les modules du serveur, tels que PHP ou SSL sont présents dans un répertoire dédié. Le serveur les invoque directement lors de son démarrage. Il appelle alors les fichiers d'extension `load` et `conf`.

```
Include /etc/apache2/mods-enabled/*.load
Include /etc/apache2/mods-enabled/*.conf
```

Le fichier nommé `ports.conf` contient la liste des ports sur lesquels le serveur devra écouter. Le serveur lit le fichier à travers la directive `Include`, tout comme pour les fichiers précédents :

```
Include /etc/apache2/ports.conf
```

À l'intérieur de ce fichier, nous retrouvons la directive `Listen` ainsi que les valeurs numériques des ports désirés :

```
Listen 80
Listen 443
```

Nous avons évoqué différents codes retours dans ce chapitre. Certains apparaissent relativement souvent, comme lorsqu'une page est demandée alors qu'elle n'existe pas, ou bien lorsque l'accès à une ressource est restreint. Il est possible de personnaliser ces pages afin de suivre la charte graphique du site, ainsi que d'avoir un rendu maîtrisé et générique. Pour cela, nous associons le code d'erreur, ici 404, à une page spécialement conçue pour l'affichage lorsque cette erreur est rencontrée :

```
ErrorDocument 404 /missing.html
```

Lorsque l'on se connecte à un site ou sur un répertoire précis, le serveur peut réagir selon différents schémas. Soit il affiche une page d'accueil, soit un listing, soit une page d'erreur, si le listing n'est pas autorisé et qu'aucune page d'accueil n'est présente. Ces pages d'accueil peuvent prendre bien des noms. L'usage veut qu'elles se nomment *index*. À l'aide de la directive `DirectoryIndex`, nous pouvons décrire l'ensemble des pages que nous considérons comme étant des pages d'accueil valides, ainsi que l'ordre dans lequel nous souhaitons les voir traitées. De ce fait, la page HTML sera traitée avant la page CGI qui sera elle-même affichée avant la page PERL, etc.

```
DirectoryIndex index.html index.cgi index.pl index.php index.xhtml
```

Pour finir la configuration de notre serveur, il ne nous reste plus qu'à charger la configuration des sites que nous souhaitons publier. Ces sites sont généralement décrits sous forme d'hôtes virtuels.

```
Include /etc/apache2/sites-enabled/
```

Notre serveur est dès à présent fonctionnel, il ne reste plus qu'à avoir des sites à publier.

Chapitre 6

Équilibrage de charge

Notre maquette présente des services web ainsi que des services DNS. Ce sont des services sur lesquels nous pouvons initialement proposer de la répartition de charge. À terme, si la plate-forme est amenée à se développer, il sera bien sûr possible d'étendre ce mécanisme à d'autres services. Nous allons présenter, dans un premier temps, les différents algorithmes parmi lesquels nous ferons notre choix pour établir notre équilibrage, puis nous présenterons dans un second temps la solution technique à mettre en place pour que le service soit opérationnel. Il est important de noter que de nombreuses solutions propriétaires existent, nous resterons dans le périmètre de notre mémoire en abordant les algorithmes proposés par le noyau Linux.

I Les différents algorithmes

Il existe un certain nombre d'algorithmes de gestion de la charge. Voici un descriptif de ceux présents dans le noyau Linux et parmi lesquels nous avons le choix pour la mise en place de notre répartiteur de charge. L'algorithme le plus fréquemment employé est le *round-robin*, ici nommé *robin-robin*, utilisé dans la fourniture d'un service web en haute disponibilité.

1. Ordonnancement robin-robin | robin-robin scheduling

On dirige la connexion selon un *round-robin* classique, c'est à dire que l'on distribue la tâche à chaque serveur de manière séquentielle, sous forme de boucle.

2. Ordonnancement robin-robin pondéré | weighted robin-robin scheduling

L'ordonnancement est effectué en fonction des poids attribués à chaque serveur de destination. Les serveurs qui se voient attribuer les poids les plus importants reçoivent les nouvelles connexions avant les serveurs ayant un poids inférieur. De fait, ils sont également plus sollicités.

3. Ordonnancement selon la connexion | least-connection scheduling

Le flux est dirigé vers le serveur qui possède le moins de connexions actives.

4. Ordonnancement pondéré selon la connexion | weighted least-connection scheduling

La requête est dirigée vers le serveur possédant la meilleure connexion active, mais également en fonction de son poids. Ainsi, pour deux connexions identiques, c'est le poids qui départagera le serveur qui recevra l'interrogation.

5. Ordonnancement selon la connexion et la répartition | locality-based least-connection scheduling

Cet algorithme se base sur les adresses IP de destination. Il est principalement utilisé en tant que *cache cluster*. Selon l'état du serveur de destination réel, à savoir s'il est actif et en sous charge ou alors en surcharge, le répartiteur de charge va envoyer le paquet directement à ce serveur ou, dans le second cas, à un serveur moins sollicité.

6. Ordonnancement avec réplication selon la connexion et la répartition | locality-based least-connection with replication scheduling

Identique à l'ordonnanceur précédent, cette solution diffère sur la fonctionnalité suivante : le répartiteur de charge maintient une cartographie d'une source vers un ensemble de serveurs, issus de la globalité du cluster. La requête est attribuée au serveur possédant la meilleure connexion de cet ensemble. Si tous les serveurs sont surchargés, il va chercher un serveur possédant la meilleure connexion dans le cluster pour l'ajouter à l'ensemble qu'il gère en relation avec notre source. Si ce groupe de serveurs n'est pas modifié au-delà d'un temps défini, le noeud le plus chargé est enlevé de l'ensemble, afin d'éviter une surcharge de la réplication.

7. Ordonnancement par les condensats de destination | destination hashing scheduling

Une table statique de condensats énumère les différents réseaux ou hôtes destinataires disponibles. Les connexions sont dirigées en fonction des entrées de cette table.

8. Ordonnancement par les condensats de source | source hashing scheduling

Dans cet algorithme, une table de condensats statique existe également, cependant elle considère la source de la connexion. L'ordonnancement est réalisé en fonction de celle-ci.

9. Ordonnancement selon le plus court délai | shortest expected delay scheduling

La connexion réseau est assignée au serveur qui possède le délai d'attente le plus court. Ce délai est calculé selon l'expression mathématique suivante :

$$\frac{C_i + 1}{U_i}$$

- avec C_i le nombre de connexion sur le serveur,
- U_i le taux de service (poids) de ce serveur.

10. Ordonnancement sans queue | never queue scheduling

C'est un modèle à deux vitesses. Lorsqu'un serveur est disponible, le travail lui est envoyé, plutôt que d'attendre la libération d'un serveur plus *rapide*, c'est à dire avec un meilleur poids. Lorsqu'aucun serveur n'est disponible, la tâche est envoyée au serveur qui minimise le délai d'attente (voir 9., ci-dessus).

II Configuration

Nous venons de voir les différents algorithmes permettant d'effectuer de l'équilibrage de charge, voire de la détection en cas de faiblesse d'un des noeuds. Comme nous le montre la figure 6.1, le client contacte uniquement le répartiteur de charge, nommé *directeur*, qui se charge de diriger le trafic, selon l'algorithme configuré, vers le serveur de destination. C'est ce serveur qui traite et répond à la demande.

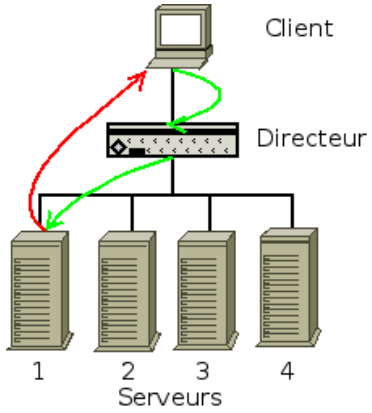


FIG. 6.1 – Exemple de fonctionnement logique du répartiteur de charge

Sur le directeur, une configuration et une compilation du noyau sont nécessaires pour pouvoir transformer un serveur linux en un répartiteur de charge. Voici les options qu'il faut activer :

```
# Cette option positionne le répartiteur de charge nommé
# IP : Virtual Server, que l'on retrouve sous le sigle IPVS
CONFIG_IP_VS=y

# Allocation de la taille de la table de connexion
# Ici, le nombre 12 correspond à 4096 entrées
CONFIG_IP_VS_TAB_BITS=12

# Support des protocoles de transport
CONFIG_IP_VS_PROTO_TCP=y
CONFIG_IP_VS_PROTO_UDP=y
CONFIG_IP_VS_PROTO_ESP=y
CONFIG_IP_VS_PROTO_AH=y

# Algorithmes d'ordonnancement
CONFIG_IP_VS_RR=y
```

```
CONFIG_IP_VS_WRR=y
CONFIG_IP_VS_LC=y
CONFIG_IP_VS_WLC=y
CONFIG_IP_VS_LBLC=y
CONFIG_IP_VS_LBLCR=y
CONFIG_IP_VS_DH=y
CONFIG_IP_VS_SH=y
CONFIG_IP_VS_SED=y
CONFIG_IP_VS_NQ=y

# Module de gestion du protocole FTP
# CONFIG_IP_VS_FTP is not set
```

Après la configuration du noyau, il faut mettre en place la configuration du directeur, puis les informations concernant chaque serveur.

Pour commencer, on autorise le trafic IP à traverser notre répartiteur :

```
#set ip_forward ON for vs-nat director (1 on, 0 off).
echo "1" >/proc/sys/net/ipv4/ip_forward
```

Cependant, seul le frontal doit répondre aux requêtes ICMP, pas les vrais serveurs. On empêche donc la redirection du trafic sur ce protocole :

```
echo "0" >/proc/sys/net/ipv4/conf/all/send_redirects
echo "0" >/proc/sys/net/ipv4/conf/default/send_redirects
echo "0" >/proc/sys/net/ipv4/conf/eth0/send_redirects
```

Il faut par la suite configurer les interfaces réseaux ainsi que le routage par défaut :

```
#setup VIP
/sbin/ifconfig eth0 192.168.0.5 broadcast 192.168.0.255 netmask 255.255.255.0

#setup private network IP
/sbin/ifconfig eth1 10.10.10.1 broadcast 10.10.10.15 netmask 255.255.255.240

#set default gateway
/sbin/route add default gw 192.168.0.254 netmask 0.0.0.0 metric 1
```

La configuration réseau de notre directeur est à présent terminée. Il faut maintenant mettre en place la partie qui concerne purement IPVS. On commence par vider la table de connexion :

```
/sbin/ipvsadm -C
```

On installe ensuite le service sur l'adresse d'entrée, sur le port HTTP (80), avec l'ordonnanceur *round robin* :

```
/sbin/ipvsadm -A -t 192.168.0.5:http -s rr
```

Il ne nous reste plus qu'à instancier les différents serveurs de destination. Ici, nous n'en décrirons que deux pour l'exemple. Le flux HTTP est transmis depuis la machine `192.168.0.5` vers le serveur

`10.10.10.11`, en utilisant le *LVS-NAT* (-m) et en attribuant une pondération de valeur 1. Le LVS-NAT est interne à IPVS, il va permettre le relais des paquets sur un plan d'adressage différent. Si nous avions souhaité conserver un plan d'adressage unique, nous aurions pu utiliser le *LVS-DR*, pour direct routing, qui a cependant une performance moindre en terme de résultat.

```
/sbin/ipvsadm -a -t 192.168.0.5:http -r 10.10.10.11:http -m -w 1
```

On effectue la même manipulation pour déclarer le deuxième serveur :

```
/sbin/ipvsadm -a -t 192.168.0.5:http -r 10.10.10.12:http -m -w 1
```

Il ne nous reste plus qu'à vérifier que le service est en place et prêt à fonctionner, pour cela, nous utilisons la commande `ipvsadm` qui reprend l'ensemble des éléments que nous avons déclarés, ainsi que les connexions actives :

```
/sbin/ipvsadm
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.0.5:www rr
  -> 10.10.10.2:www               Masq    1      0          0
  -> 10.10.10.3:www               Masq    1      0          0
```

La dernière et ultime étape consiste à configurer les paramètres réseaux sur chaque serveur :

```
/sbin/ifconfig eth1 10.10.10.X broadcast 10.10.10.15 netmask 255.255.255.240
route add default gw 10.10.10.1
```

Afin de tester le fonctionnement de notre montage, nous avons configuré une page différente pour chaque serveur pour de déterminer si la répartition de charge s'effectue correctement. Cela fonctionne sans aucun problème, il nous reste simplement à mettre en place le site Internet de l'école et à nous assurer que l'on passe bien d'un serveur à l'autre sans que l'utilisateur final ne puisse se rendre compte du saut :

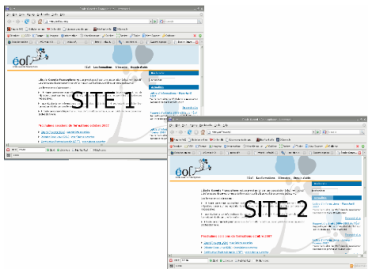


FIG. 6.2 – Site Internet de l'Éof sur deux serveurs répartis

Chapitre 7

Le Shell Sécurisé

I Préambule

Il y a aujourd'hui deux possibilités de travail, en local (local work) ou à distance (remote work). Ce dernier est permis par l'utilisation du réseau, qu'il soit local (LAN) ou réellement éloigné, comme par Internet (WAN). De nombreux outils ont été fournis pour utiliser la capacité du réseau. Échanger, copier, utiliser des shells à distance. Les noms de ces outils sont respectivement FTP, rcp, telnet, etc... Bien que ces outils, utilisés pendant des années et même encore aujourd'hui dans certaines entreprises, soient très pratiques, ils comportent une faiblesse importante. Leurs transactions sont transmises en clair via le réseau. De ce fait, l'informatique devenant ouverte à un grand nombre, n'importe quelle personne mal intentionnée peut être en mesure d'observer ce que vous faites, allant même jusqu'à subtiliser vos données personnelles et mots de passe. Bien que les techniques sur les réseaux locaux évoluent, par exemple en prévenant le spoofing *ARP*, le risque reste présent, quant aux communications distantes, la seule sécurisation possible viendra de l'éducation de l'utilisateur final, ainsi que de la mise en place de techniques cryptographiques. Le *Shell Sécurisé*, plus communément appelé *ssh* est donc une réponse à cette problématique.

1. Le problème par la pratique

Afin de pouvoir appuyer cette notion de non-confidentialité, nous allons regarder ce qui se passe lors de l'établissement d'une connexion d'une session ftp. Le logiciel utilisé pour tracer les connexions est Wireshark [1]. Établissons la connexion sur un serveur ftp fonctionnant sous GNU/Linux. Après quelques échanges de paquets, nous obtenons la bannière d'accueil :

on constate que la connexion s'effectue sur le port 21 (en 22 et 23 dans la trame, lire *Ox15*). Les données quant à elles commencent clairement en 42, avec la bannière d'accueil :

```
220 Serveur de mise à jour des pages perso
```

Les données ont été envoyées en clair. Le mot de passe est ici clairement exprimé. On peut lire *PASS test* :

Nota : Notre étude ne se portant pas sur le service ftp, nous ne rentrerons pas sur le détail des transmissions et considérons que ces quelques remarques suffisent pour notre étude.

[1] http ://wireshark.org/

```
0000  00 0b cd a4 5f 63 00 60  4c c7 d8 68 08 00 45 00   ...._c.` L..h..E.
0010  00 8a 6b cf 40 00 35 06  e7 cc d4 1b 3f 03 c0 a8   ..k.@.5. ....?...
0020  1e 0b 00 15 0e 02 49 21  f5 9d 2d 3b b5 2f 80 18   ......I! ..-;./..
0030  05 a8 16 73 00 00 01 01  08 0a d1 b6 9e 52 00 79   ...s.... .....R.y
0040  c7 14 32 32 30 20 53 65  72 76 65 75 72 20 64 65   ..220 Se rveur de
0050  20 6d 69 73 65 20 61 20  6a 6f 75 72 20 64 65 73    mise a  jour des
0060  20 70 61 67 65 73 20 70  65 72 73 6f 20 64 65 20    pages p erso de
0070  46 72 65 65 2e 66 72 20  76 65 72 73 69 6f 6e 20   Free.fr  version
0080  5b 4a 61 6e 20 33 30 20  32 30 30 36 20 31 37 3a   [Jan 30  2006 17:
0090  32 38 3a 30 34 5d 0d 0a                           28:04]..
```

FIG. 7.1 – Bannière FTP

```
0000  00 60 4c c7 d8 68 00 0b  cd a4 5f 63 08 00 45 10   .`L..h.. .._c..E.
0010  00 3f 73 fd 40 00 40 06  d4 d9 c0 a8 1e 0b d4 1b   .?s.@.@. ........
0020  3f 03 0e 02 00 15 2d 3b  b5 49 49 21 f6 59 80 18   ?.....-; .II!.Y..
0030  05 b4 6e e3 00 00 01 01  08 0a 00 79 c7 f2 d1 b6   ..n..... ...y....
0040  9e 7d 50 41 53 53 20 74  65 73 74 0d 0a            .}PASS t est..
```

FIG. 7.2 – Mot de passe en clair sur une authentification FTP

De la même manière, tout comme on vient de dérober le mot de passe, on peut récupérer les conversations, les fichiers. La confidentialité est rompue, la porte est ouverte à l'insertion inopportune de personnes non autorisées dans le système…. Si l'échange présenté relate une session FTP, il aurait pû en être de même avec une session telnet. Cependant, le protocole est lourd, cet échange a donc été préféré.

II La solution proposée

Afin de ne plus être ouvert au monde entier, nous avons la possibilité de chiffrer nos échanges. Pour ce faire, nous allons installer le couple client/serveur ssh, Openssh. Outil libre et gratuit. Présent sur un serveur, l'outil permet de joindre tout autre serveur équipé d'un serveur ssh et d'être joint par les clients supportant le même protocole. De plus, de nombreux outils d'échange et de sauvegarde se basent sur ce couple, outils que nous présenterons par la suite.

Afin de mieux comprendre le comportement et l'utilité de chaque programme, décomposons les différents paquets ainsi que l'usage fait des exécutables.

- /usr/sbin/sshd : serveur ssh
- /usr/bin/scp : copie distante sécurisée
- /usr/bin/ssh-keygen : génération de clés d'authentification, management et conversion
- /usr/bin/sftp : transfert sécurisé de fichiers
- /usr/bin/slogin : client ssh
- /usr/bin/ssh : client ssh
- /usr/bin/ssh-add : ajoute les identités DSA ou RSA à l'agent d'authentification
- /usr/bin/ssh-agent : agent d'authentification
- /usr/bin/ssh-keyscan : recueille les clés publiques ssh

Nous traiterons chacune de ces applications afin de comprendre dans quels contextes elles peuvent être utilisées.

III Connexion à un hôte

Le client ssh opère selon un ordre défini :

1. les paramètres en ligne de commande
2. les informations spécifiques à l'utilisateur du fichier ~/.ssh/config (voir annexe E)
3. la configuration globale du système dans le fichier /usr/local/etc/ssh_config ou /etc/ssh/ssh_config (cela dépendant de la distribution utilisée)

Il existe différentes options pour se connecter à un hôte via ssh.
- -l login : Identifiant de l'utilisateur.
- -v -vv -vvv : Mode verbeux, permet d'obtenir les messages de debugage plus ou moins complets (plus il y a de v, plus vous obtenez d'informations, le nombre maximum étant 3).
- -1 ou -2 : version de ssh employée. Il est déconseillé d'employer la version 1 du protocole. Bien qu'aucun exploit public ne circule, les faiblesses cryptographique du protocole ont été prouvées. De plus, la version 2 est reconnue par l'IETF
- -p port : Numéro du port distant

Exemple d'utilisation :

```
ssh -vv -l utilisateur -p port -(1|2) hôte
```

ou

```
ssh utilisateur@hôte
```

L'hôte distant doit avoir un service ssh, nommé sshd, qui permet la connexion.

IV Création de paires de clés

Ssh s'appuie sur des algorithmes à paire de clés, ce qui signifie que vous disposez d'une clé publique, disponible pour tout un chacun et une clé privée dont on garde jalousement l'entrée. Ce système va nous permettre de nous authentifier auprès des hôtes que nous désirons contacter. Il nous faut au préalable créer le trousseau. Pour cela, il existe deux implémentations réalisables : DSA ou RSA. La version 2 du protocole supporte les deux méthodes de chiffrement alors que la version première accueillait uniquement RSA. Le choix, de ce fait, est laissé à l'appréciation de l'utilisateur, avec la possibilité donnée à l'administrateur, de forcer l'utilisation de l'une ou l'autre des méthodes.

```
$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/jop/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jop/.ssh/id_dsa.
Your public key has been saved in /home/jop/.ssh/id_dsa.pub.
The key fingerprint is:
4a:0b:3b:eb:ed:05:47:56:cb:23:28:d3:d7:81:69:08

$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jop/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

```
Your identification has been saved in /home/jop/.ssh/id_rsa.
Your public key has been saved in /home/jop/.ssh/id_rsa.pub.
The key fingerprint is:
52:65:28:9a:8b:64:cb:b7:6e:70:75:10:d9:0a:01:d9
```

Pour les deux algorithmes (dsa, rsa), le système nous demande dans quel fichier nous désirons sauvegarder la clé. Les fichiers par défaut semblent une bonne solution. Par la suite, une passphrase nous est demandée. Celle-ci est un « mot de passe amélioré », car non limité à un mot ou une petite suite de caractères. Il faut cependant prendre des précautions, car en cas de perte de la passphrase, vous ne pourriez plus vous authentifier en tant que propriétaire authentique.

Nous allons maintenant voir trois méthodes de connexion via ssh.

1. Connexion par mot de passe

La première des méthodes, la plus connue et la plus usitée : l'hôte distant demande un mot de passe pour s'assurer de votre identité.

```
$ ssh -p 222 -l root 192.168.0.1
root@192.168.0.1's password:
Last login: Thu Dec 26 03:37:03 2006 from centaur
$
```

Je me connecte au port 222 sur une machine de mon réseau local, demandant de m'identifier comme root. La machine n'est pas connue dans mon fichier known_hosts. Ce fichier contient les clés hôte DSA des serveurs SSH auxquels l'utilisateur s'est connecté. Cette méthode de connexion est intéressante, mais minimise les capacités intrinsèques de ssh. Pour des connexions telles que, vers un serveur CVS, un tunnel pour le serveur de courrier, il serait fastidieux de s'authentifier à chaque fois par ce moyen.

2. Connexion par paires de clés

Puisque nous utilisons des algorithmes à paire de clés (rsa, dsa), c'est à dire composée d'une clé secrète et d'une clé publique, il faut bien que cela nous serve à quelque chose. Nous allons donc automatiser la connexion. Pour ce faire, l'hôte contient un fichier authorized_keys dans le répertoire .ssh où nous nous connectons (en général un home directory). Il suffit de copier l'identifiant ou les identifiants des clés publiques pour que nous soyons reconnus du serveur.

Attention, il faut que PubkeyAuthentication soit positionné à yes dans le fichiers de configuration du serveur. Pour insérer notre clé, plusieurs méthodes sont à notre disposition. nous pouvons employer des moyens conventionnels tels que le ftp ou le mail (à l'administrateur par exemple), ou nous pouvons le faire au moyen d'outil sécurisé. Puisque c'est notre sujet, profitons-en pour donner un exemple de scp sur lequel nous reviendrons ultérieurement.

```
scp .ssh/id_dsa.pub jop@faith:/home/jop/.ssh/dsa2connex
Warning: Permanently added 'faith' (RSA) to the list of known hosts.
jop@faith's password:
id_dsa.pub 100% |*****************************| 613 00:00
```

Le fichier est maintenant copié sur l'hôte distant, il reste à inclure la clé dans le fichier authorized_keys. Une simple commande suffira sur le serveur ssh :

```
cat dsa2connex >>authorized_keys
```

Nous pouvons maintenant nous connecter, l'hôte distant nous reconnaît. Nous pouvons nous connecter sans mot de passe et avec une passphrase, si nous en avons au préalable déclaré une lors de la création de la paire de clés :

```
$ ssh -p 222 -l root 192.168.0.1
Last login: Thu Dec 26 03:37:03 2006 from centaur
$
```

3. L'agent d'authentification

Nous savons maintenant nous connecter à un hôte distant connaîssant notre identité par l'intermédiaire de notre clé publique. Nous avons vu précédemment que nous étions libres de mettre, ou non, une passphrase afin de compliquer l'usurpation qui pourrait en être faite. S'il paraît fastidieux d'insérer un mot de passe à chaque connexion, cela l'est d'autant plus lorsque l'on doit rentrer une phrase entière. L'agent ssh est là pour nous simplifier la vie. Lancé au début de la session (terminal ou graphique), il retient la passphrase le temps où ssh-agent sera actif (le temps, par exemple, d'une session). Nous proposons ici deux modes d'utilisation de cet agent, cependant, celui-ci n'est limité que par l'utilisation que l'on en a. Il est tout à fait possible de lier un shell ou une console virtuelle à ce dernier.

Pour une session en mode terminal :

```
$ssh-agent screen
$ssh-add
```

Après avoir lancé l'agent ssh, on ajoute la passphrase à l'agent par l'intermédiaire de ssh-add ; tous les hôtes distants utilisant le protocole ssh, ou un de ses outils dérivés, et disposant de notre clé publique ne nous demanderont plus la passphrase puisque gérée par l'agent ssh.

De même, en mode graphique :

```
$ssh-agent startx
```

Il suffit ensuite d'ouvrir un terminal et de taper :

```
$ssh-add
```

L'agent sera actif pour toutes les applications utilisées en mode graphique.

Une dernière méthode consiste à lancer l'agent-ssh qui fournit un certain nombre de variables à exporter, puis demander l'action de `ssh-add` puis `ssh-agent`

On récupère les variables fournies et on les exporte :

```
$SSH_AUTH_SOCK=/tmp/ssh-XXy0hiXW/agent.2019; export SSH_AUTH_SOCK;
$SSH_AGENT_PID=2020; export SSH_AGENT_PID;
$echo Agent pid 2020;
```

On ajoute : $ssh-add

Pour tuer l'agent ssh, il suffit de faire :

```
$ ssh-agent -k
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
echo Agent pid 2020 killed;
```

4. Invalidité des clés d'un hôte connu

Comme nous l'avons vu précédemment, lors de la connexion, il nous est possible d'enregistrer l'empreinte (fingerprint) fournit par le serveur distant. Dans le cas où celle-ci serait modifiée, nous serions immédiatement prévenus. Cela peut découler de quatre choses :

- Un piratage a permis la modification des clés présentes dans votre known_hosts.
- Le serveur distant s'est fait pirater et la valeur de ses clés a changé.
- La troisième peut correspondre à une attaque en bonne et due forme d'un man-in-the-middle.
- La dernière et néanmoins plus plausible des solutions nécessite que l'administrateur du serveur distant ait changé les clés.

La seule solution fiable demande de contacter l'administrateur en question et lui demander ce qu'il en est. Dans la pratique, peu de personnes usent de cette assurance de sécurité.

```
$ ssh 127.0.0.1
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
8b:d5:26:a3:79:ed:25:0f:3b:6f:fe:30:1f:ed:89:ae.
Please contact your system administrator.
Add correct host key in /home/jop/.ssh/known_hosts to get rid of this message.
Offending key in /home/jop/.ssh/known_hosts:13
RSA host key for 127.0.0.1 has changed and you have requested strict checking.
Host key verification failed.
```

Dans le cas où le changement est validé par l'administrateur du serveur distant, il suffira de supprimer la ligne correspondante dans le fichier *known_hosts* pour pouvoir nous reconnecter.

V La copie sécurisée

Comme nous avons pu le voir plus haut, ssh fournit un outil de copie sécurisée en standard, sous le nom de scp pour SecureCoPy. Il remplace son ancêtre rcp. Son usage est très simple :

scp hôte_d_ou_je_veux_copier :source_copie hôte_destination :cible

Il n'est pas nécessaire de préciser l'hôte lorsque celui-ci correspond à l'hôte où l'on se trouve.

```
$scp iptables faith:/home/jop/download
iptables 100% |*****************************| 27654 00:00
```

Il est également possible de faire des copies récursives, comme nous le ferions avec n'importe quel autre utilitaire de copie.

```
$scp -r download faith:/home/jpg/dl
img2.png 100% |*****************************| 277 00:00
internals.pl 100% |*****************************| 188 00:00
labels.pl 100% |*****************************| 271 00:00
images.pl 100% |*****************************| 624 00:00
portscanning.css 100% |*****************************| 891 00:00
index.html 100% |*****************************| 56753 00:00
```

VI Le transfert de fichier sécurisé

Tout comme on peut copier des fichiers à distance par l'intermédiaire de scp, il est également possible de transférer des fichiers par l'intermédiaire d'un ftp sécurisé nommé SecureFTP. Bien que l'outil soit encore limité (pas de reprise de chargement en cas de coupure, pas d'indication du temps restant, pas de telechargement ou de chargement récursif, ...), il permet de faire toutes les manipulations basiques disponibles sur un ftp non sécurisé.

Le tableau 7.1 reférence la liste exhaustive des commandes disponibles sur sftp.

À la manière d'un FTP classique, il suffit, pour se connecter, d'indiquer son login et l'hôte à joindre.

```
$ sftp jop@faith
Connecting to faith...
$sftp> lpwd
Local working directory: /home/jop
$sftp> lcd download
$sftp> lpwd
Local working directory: /home/jop/download
$sftp> put iptables
Uploading iptables to /home/jop/iptables
$sftp> quit
```

VII Le tunnel et le Xforwarding

Tout au long de ce chapitre, nous avons pu découvrir des applications connues, il nous manque cependant l'exportation des applications distantes. En effet, à l'aide de telnet, nous pouvions utiliser un logiciel non présent sur votre machine, mais présent sur l'ordinateur distant. Il suffisait de taper :

```
$export DISPLAY=mon_adresse:0.0
```

Comme par magie, l'application arrivait plus ou moins rapidement selon la connexion sur notre écran. C'est ce qu'on appelle du Xforwarding. L'avantage une fois de plus d'utiliser ssh réside dans la connexion chiffrée et l'impossibilité à un agresseur éventuel de lire ce que nous faisons via le réseau.

TAB. 7.1 – Liste des commandes sftp

cd *path*	Change le répertoire distant vers 'path'
lcd *path*	Change le répertoire local vers 'path'
chgrp grp *path*	Change le groupe du fichier 'path' par 'grp'
chmod mode *path*	Change les permissions du fichier 'path' à 'mode'
chown own *path*	Change le propriétaire du fichier 'path' par 'own'
help	Affiche ce message d'aide
get *remote-path* [*local-path*]	Télécharge le fichier
lls [ls-options [*path*]]	Affiche le listing du répertoire local
ln *oldpath newpath*	Crée un lien symbolique du fichier distant
lmkdir *path*	Crée un répertoire local
lpwd	Affiche le répertoire courant
ls [*path*]	Affiche le listing du répertoire distant
lumask umask	Positionne l'umask local à 'umask'
mkdir *path*	Crée le répertoire distant
put *local-path* [*remote-path*]	Charge le fichier
pwd	Affiche le répertoire courant distant
exit	Quitte sftp
quit	Quitte sftp
rename *oldpath newpath*	Renomme le fichier distant
rmdir *path*	Supprime le répertoire distant
rm *path*	Supprime le fichier distant
symlink *oldpath newpath*	Crée un lien symbolique du fichier distant
version	Affiche la version de sftp
!command	Exécute la 'commande' dans un shell local
!	Sort vers un shell local
?	Affiche ce message d'aide

```
$ssh -X -C jop@faith gedit
```

Dans l'exemple donné, nous demandons d'ouvrir la connexion ssh pour inscrire Gedit à l'intérieur (bloc-notes de Gnome). En fermant Gedit, nous fermerons la connexion ssh. Pour que cela soit réalisable, il faut cependant activer l'option X11Forwarding dans les fichiers de configuration du serveur. Les options *X* et *C* du client ssh permettent la demande d'activation du Xforwarding s'il est autorisé, ainsi que la compression des données, permettant un meilleur rendement sur le tunnel établi.

Toujours dans la même idée de tunneler des applications, ssh nous permet d'encapsuler des protocoles. Cela est très intéressant lorsque nous avons recours à des protocoles tels que smtp, pop, ... Nos courriers, par exemple, ne seront plus à la merci de fournisseurs indiscrets, ou des fouines du réseau.

Encore une fois, ssh implémente par défaut cette possibilité ; pour ce faire, voilà la syntaxe proposée :

```
ssh -L port_local:hôte_distant:port_distant nom_utilisateur@nom_hôte
```

Admettons que l'on veuille encapsuler les ports 25 (smtp) et 110 (pop), il nous suffira d'exécuter les lignes suivantes :

```
$ssh -L 2025:faith:25 -L 2110:faith:110 jop@faith
```

Il faut par la suite configurer le logiciel de messagerie favori aux ports 2025 pour le smtp et 2110 pour le pop, l'adresse étant celle du serveur local, afin de recevoir correctement le courrier, et ce de manière sécurisée, par l'intermédiaire de notre tunnel. Les ports inférieurs à 1024 sont réservés à l'usage de l'administrateur. Il est nécessaire de prendre des ports dans la tranche allant de 1024 à 65535.

Il existe deux autres types de tunnels. Le premier consiste dans l'établissement d'un *remote shell* qui permet la mise en place d'un tunnel sur l'hôte distant avec la machine locale. Lorsque le port du tunnel configuré sur la machine distante sera interrogé, il contactera directement la machine locale. Le deuxième est un VPN directement construit sur le protocole SSH.

VIII Autre implémentation du protocole

Openssh est l'implémentation la plus connue, mais n'est pas la seule. On peut également utiliser :

- Lsh, une version GNU.
- Dropbear, un serveur dédié à l'embarqué.
- Conch, une implémentation en python.

IX Implémentation au sein de la maquette

Nous avons vu dans ce chapitre les différents usages possibles apportés par *ssh*. Pour ce projet, nous nous intéressons principalement à la commande ssh qui permet de nous authentifier à distance ainsi que d'intervenir par l'intermédiaire d'un shell distant. Nous utilisons également la copie sécurisée par l'intermédiaire de la commande *scp*. Dans ce cas, puisque des scripts sont les initiateurs, et afin qu'aucun mot de passe ne soit stocké en clair sur le système, des clés sans passphrases sont insérées dans les clés autorisées des systèmes distants, permettant une copie directe et authentifiée. Bien sûr, ces automatisations doivent être restreintes et surveillées, car en cas de piratage, c'est une porte ouverte inter-serveur. C'est pourquoi, nous complétons chaque compte utilisé par l'emploi d'un shell restreint, rssh.

1. rssh

Derek Martin a écrit *rssh* afin de combler le manque d'implémentation fournie au niveau de *ssh* en ce qui concerne les prisons utilisateurs. Celles-ci permettent de laisser à disposition un espace de stockage ou de manipulation pour l'utilisateur, sans qu'il ne puisse jamais influer sur le système et sa sécurité. En effet, il est impossible, de manière simple et satisfaisante, d'enfermer un utilisateur dans un répertoire ou de restreindre ses actions. L'implémentation commerciale de *ssh* apportait la fonctionnalité, Derek l'a fait vivre pour la partie libre.

Techniquement il s'agit d'un petit binaire qui se place dans `/usr/bin/rssh`, et dont le fichier de configuration se loge dans `/etc/rssh.conf`. Pour le reste, ce n'est que de la configuration que nous allons détailler ci-après. Le logiciel permet de gérer différents outils : scp pour la copie sécurisée, sftp pour le transfert sous forme ftp, cvs ou subversion pour la gestion concurrente de versions, rdist permet le maintien de copies identiques sur différents systèmes et enfin rsync permet la synchronisation distante, autorisant, de fait, différentes possibilités de sauvegardes (incrémentales ou totales). Il interdit la connexion par shell. Pour chaque utilisateur, il sera alors possible de décider quel service précis sera autorisé.

Pour activer rssh, il faut au préalable changer le shell par défaut de l'utilisateur et lui adjoindre le shell particulier de *rssh* :

```
jop:x:1000:1000:,,,:/home/jop:/bin/bash
```

deviendra

```
jop:x:1000:1000:,,,:/home/jop:/usr/bin/rssh
```

Dans chaque dossier personnel, il faut ensuite insérer l'ensemble de ces fichiers nécessaires au fonctionnement de la prison, c'est à dire les dossiers, les binaires et les librairies correspondantes :

```
usr
|-- bin
|   |-- rssh
|   `-- scp
`-- lib
    |-- i686
    |   `-- cmov
    |       `-- libcrypto.so.0.9.7
    |-- rssh
    |   `-- rssh_chroot_helper
    `-- sftp-server

lib
|-- ld-linux.so.2
|-- libc.so.6
|-- libcrypt.so.1
|-- libdl.so.2
|-- libnsl.so.1
|-- libresolv.so.2
|-- libutil.so.1
`-- libz.so.1
```

Pour chaque utilisateur, il faut ensuite insérer une ligne dans `/etc/rssh.conf` :

```
user=username:umask:access bits:path
```

La ligne se décompose avec les paramètres suivants :

username : le nom de l'utilisateur pour qui l'entrée fournit les options

umask : l'umask de l'utilisateur, tel qu'il doit être spécifié dans le shell

access bits : cinq chiffres binaires qui indiquent si l'utilisateur a le droit d'utiliser rsync, rdist, cvs, sftp, et scp, dans cet ordre. Le *1* autorise la commande alors que le *0* l'interdit.

path : répertoire cible dans lequel l'utilisateur devra être emprisonné.

Voici un exemple de ce que pourrait être une configuration authentique, autorisant uniquement le *scp* :

```
user=jop:022:00001:/home/jop
```

Toutes ces actions de configuration peuvent bien sûr être rassemblées dans un script qui automatiserait le process.

X Filtrage actif des connexions

Le protocole SSH est devenu le digne remplaçant du protocole telnet en quelques années seulement. Son lot de services annexes ne fait que confirmer cette suprématie. Tout succès ayant un revers, SSH n'échappe pas à la règle. En effet, de nombreuses attaques par dictionnaire viennent remplir les archives des services SSH. Des logiciels tentent une authentification à l'aide d'un identifiant et d'un mot de passe. Si cette tentative est un succès, l'utilisateur obtient un compte qu'il pourra utiliser à loisir, si c'est un échec, un autre couple est utilisé, et ce jusqu'à ce que la liste fournie au logiciel (le dictionnaire) soit vide.

Face à ces tentatives, bien souvent veines dans le cadre d'une politique de gestion des mots de passe cohérente, il y a deux possibilités :

– laisser faire,

– bannir les IPs des fautifs.

L'authentification SSH, en cas de succès comme en cas d'échec, est archivée. Si l'on décide de ne rien faire, les fichiers prennent du volume rapidement. Les archives deviennent inutilisables et de la place est gaspillée inutilement. Si l'on souhaite surveiller ces tentatives et agir en conséquence, nous pouvons soit écrire notre propre script, soit utiliser un logiciel comme *DenyHost*.

DenyHost a été écrit en réponse directe à une recrudescence de ces tentatives. Il surveille les archives d'authentification du service SSH. En fonction de la qualité du compte attenté, à savoir le compte administrateur, un compte valide ou un compte invalide, et de l'échec de l'authentification, Denyhost transmet l'adresse IP de la machine en faute au pare-feu, qui interdira toute nouvelle tentative.

Chapitre 8

Sécurisation des serveurs

Ce chapitre se veut transverse à tous les autres, puisqu'il propose un ensemble de manipulations sur les différents serveurs qui composent notre infrastructure. Nous n'aborderons pas ici le système de journalisation, ni le pare-feu, puisqu'ils sont décrits dans des chapitres spécifiques. Les différentes explications présentées peuvent également être valables pour tout serveur qui serait mis en production. C'est donc un ensemble élémentaire de consignes de sécurité, de bon sens, mais aussi de technique.

I Sécurisation du BIOS

Cette partie diffère quelque peu de l'ensemble du mémoire. En effet, nous abordons le seul aspect logiciel non libre, puisque intégralement lié à l'environnement matériel. Cependant, pour assurer une sécurité de bout en bout, il nous faut voir quelques éléments. Commençons par le démarrage des périphériques. Les bios actuels permettent de choisir le média initial, souvent à choisir entre le lecteur cd, la disquette, le réseau, l'usb et le disque dur. Lorsque le système est installé, nous positionnons le disque dur principal, c'est à dire celui qui contient le système à amorcer, comme périphérique à amorcer par défaut. En effet, si ce n'est pas le cas, n'importe qui ayant accès physiquement aux machines, peut charger un OS différent, par l'intermédiaire d'un *CD live* et accéder aux partitions, voire changer les données, telle que le mot de passe de l'administrateur. Lorsque cette première étape est réalisée, il nous faut affecter un mot de passe qui bloque à quiconque l'entrée de l'administration du bios et qui ne serait pas détenteur de ce dernier. Pour ce qui concerne le BIOS à proprement parlé, nous désactivons la notion de *wake on lan* qui permet un allumage du serveur à distance. Il ne reste plus, pour les disques durs et BIOS qui en ont la capacité parmi nos serveurs, qu'à affecter un mot de passe pour la lecture de ce dernier. Ainsi, en cas de vol du disque, les données ne pourront être lues que si l'utilisateur connaît le mot de passe. Dans le cas contraire, le disque refusera l'accès aux données.

II Le chargeur de démarrage

Le chargeur de démarrage est le premier logiciel que nous rencontrons lors du démarrage de l'ordinateur. Il s'occupe de la gestion de l'amorce des différents systèmes d'exploitation, ainsi que les modes associés. Il permet, par exemple, de choisir entre deux versions de noyaux proposées, ou de choisir le mode de récupération (*single mode*). Pour notre maquette, nous avons choisi l'utilisation de *GRUB*, par opposition à *Linux Loader*, puisqu'il propose un shell interactif puissant, des commandes plus poussées,

la possibilité de créer un condensat de son mot de passe, mais également parce qu'il a été choisi et inséré comme choix d'excellence par les développeurs Debian dans leur distribution. Lors de son installation, sa configuration par défaut laisse son shell ouvert aux manipulations. Il nous est donc nécessaire d'interdire l'usage de ce dernier, sauf si c'est l'administrateur du système qui y accède. Pour cela, nous invoquons le shell par la commande *grub* et nous demandons la création d'un *md5* crypté :

```
grub> md5crypt

Password: ********
Encrypted: $1$LGtpre$GJa2DwLtpwE6reisvZgtp0
```

Il ne nous reste plus qu'à insérer cette chaîne de caractères dans le fichier de configuration de grub, qui se trouve dans `/boot/grub/menu.lst`, précédé de la commande `password` et de son option `--md5`, ce qui indique au logiciel que le mot de passe est crypté.

```
password        --md5 $1$LGtpre$GJa2DwLtpwE6reisvZgtp0
```

Toute tentative d'utilisation du shell interactif au démarrage du serveur est dorénavant liée à la connaissance du mot de passe précédemment généré.

Nous devons maintenant nous préoccuper des entrées qui pourraient permettre à un utilisateur malveillant d'obtenir des droits supplémentaires. C'est le cas de l'entrée qui autorise directement le lancement du système en mode récupération. Dans ce mode, seule la racine est montée, afin de permettre une vérification des systèmes de fichiers, par exemple. Deux possibilités s'offrent à nous, par l'intermédiaire de la commande `password` de nouveau, ou par celui de la commande `lock`. Avec cette dernière commande, c'est le mot de passe précédemment créé qui permettra l'accès. De ce fait, nous choisissons d'employer de nouveau la commande `password`. Dans les deux cas, l'instruction se place de toute façon en dessous de la ligne de titre (*title*) de l'entrée à démarrer :

```
title           Debian GNU/Linux, kernel 2.6.18.1 (recovery mode)
password        --md5 $pE$wI2OfkPa486FFJCyI0F97
root            (hd0,0)
kernel          /vmlinuz-2.6.18.1 root=/dev/sdb1  ro single
savedefault
boot
```

III Gestion des comptes

Lors de l'installation du système, un certain nombre d'utilisateurs système sont déclarés par défaut. Sur une distribution correctement conçue, cela ne devrait normalement pas poser de soucis, cependant, il est préférable de s'en assurer. Ainsi, les comptes systèmes ne doivent pas pouvoir se connecter avec un mot de passe, ni dans la plupart des cas avoir un shell de connexion. Bien que la première parade doive être suffisante, il n'est pas superflu d'instaurer la deuxième. Pour cela, il faut procéder à la suppression des comptes inutiles (*games* est un bon exemple), puis éditer le fichier */etc/shadow* et placer un **!** ou une ***** dans le champ correspondant au mot de passe, ce qui aura pour effet d'annuler la possibilité de s'authentifier. L'étape suivante consiste à modifier le dernier champ du fichier */etc/passwd* correspondant au shell de connexion, en général */bin/bash* par */bin/false*, si ce dernier est correctement déclaré dans le

fichier */etc/shells*.

Au départ, notre fichier de mot de passe ressemble à ceci :

```
# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
Debian-exim:x:102:102::/var/spool/exim4:/bin/false
identd:x:100:65534::/var/run/identd:/bin/false
sshd:x:101:65534::/var/run/sshd:/bin/false
jop:x:1000:513:jop,,,:/home/jop:/bin/bash
```

Nous supprimons donc les entrées superflues et nous modifions les shells des comptes qui ne doivent pas pouvoir se logguer :

```
# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/false
bin:x:2:2:bin:/bin:/bin/false
sys:x:3:3:sys:/dev:/bin/false
sync:x:4:65534:sync:/bin:/bin/sync
man:x:6:12:man:/var/cache/man:/bin/false
lp:x:7:7:lp:/var/spool/lpd:/bin/false
mail:x:8:8:mail:/var/mail:/bin/false
www-data:x:33:33:www-data:/var/www:/bin/false
backup:x:34:34:backup:/var/backups:/bin/false
nobody:x:65534:65534:nobody:/nonexistent:/bin/false
sshd:x:101:65534::/var/run/sshd:/bin/false
jop:x:1000:1000:Jean-Philippe Gaulier,,,:/home/jop:/bin/bash
```

Nous nous assurons également que le fichier `/etc/shadow` est correctement configuré :

```
# sudo cat /etc/shadow
root:$1zdPznwd2zn$aqCssdzhzSOfrrpzmdr0:13478:0:99999:7:::
daemon:*:13478:0:99999:7:::
bin:*:13478:0:99999:7:::
sys:*:13478:0:99999:7:::
sync:*:13478:0:99999:7:::
man:*:13478:0:99999:7:::
lp:*:13478:0:99999:7:::
mail:*:13478:0:99999:7:::
www-data:*:13478:0:99999:7:::
backup:*:13478:0:99999:7:::
nobody:*:13478:0:99999:7:::
sshd:!:13478:0:99999:7:::
jop:$1$gAfefeffeDCjfdyEferYEzGRLBe1:13480:0:99999:7:::
```

IV Politique de mot de passe

L'administration d'un parc de serveurs ne peut se faire sans une bonne gestion de la politique des mots de passe. Ceci sous-entend deux choses :

– les mots de passe doivent suivre une politique de construction,
– ils doivent être changés régulièrement.

Dans cette section, nous proposons un petit état de l'art du choix d'un mot de passe.

1. Construction d'un bon mot de passe

Un bon mot de passe est un mot de passe que l'on ne peut pas trouver en utilisant d'un dictionnaire, même si ce dernier se voit doté d'astuces basiques de substitution. Cela sous-entend qu'il ne fait référence à aucun mot existant, et ce dans aucune langue. Pourquoi donc ? Parce qu'une personne mal intentionnée qui pourrait obtenir la liste des mots de passe cryptés garde la possibilité d'effectuer une attaque par dictionnaire. Donc, si son dictionnaire comporte le mot, il y a de grandes chances, avec du temps, que ce mot de passe devienne obsolète.

Ainsi, il est essentiel de choisir convenablement son mot de passe, par exemple une phrase qui nous est connue :

```
Ne manger rien ou jeûner, voilà bien votre grande bêtise !
```

Nous dépassons la première barrière inéluctable des *huit caractères*, puis la seconde des *treize*, avec des majuscules et des minuscules ainsi que des caractères de ponctuation. On pourrait considérer cette phrase comme un mot de passe acceptable. Cependant certains systèmes n'accepteraient pas une chaîne de 59 caractères pour cause de longueur excessive. En effet, certains systèmes ne vous autorisent pas à dépasser une longueur de 32 caractères pour un mot de passe. D'autres, plus anciens encore, n'acceptaient qu'un maximum de huit ou quatorze. Dans ce cas, nous avons la possibilité de garder votre phrase, mais en ne prenant par exemple que la première lettre, ce qui donnerait ce résultat :

```
Nmroj,vbvgb!
```

Il est à peu près sûr qu'aucun dictionnaire ne soit en mesure de trouver quelque chose ainsi constitué ! Il ne faut pas hésiter à mélanger des lettres majuscules et minuscules, des chiffres et des caractères spéciaux. Un très bon exemple de mot de passe serait le suivant :

```
1m,1^^at1C!
Traduction : un matin, un lapin a tué un chasseur !
```

Attention, toutefois, de nombreux systèmes d'authentification, entre autre sur Internet n'acceptent que l'emploi de certains caractères.

On peut noter qu'il existe des générateurs aléatoires de mots de passe. Si l'on souhaite les utiliser, il faut s'assurer que la graine d'entropie sur laquelle ils se basent est suffisamment *aléatoire* pour ne pas pouvoir être reproduite.

2. Périodicité d'un mot de passe

Actuellement, on considère que la durée de vie d'un mot de passe ne peut excéder trois mois. Cette extrapolation se base sur le temps minimum qu'il faudrait à une personne mal intentionnée pour venir à bout de notre mot de passe à l'aide d'outils dédiés à cet usage (table rainbow, *john the ripper*, ...). Ces outils sont disponibles aux attaquants mais sont également à notre portée. Il est donc intéressant de faire tourner ces outils sur notre station de travail pour s'assurer de la solidité des mots de passe choisis :

```
#john /home/jop/save/shadow
Loaded 3 passwords with 3 different salts (FreeBSD MD5 [32/32])
toto             (toto)
```

V Réplication des données

Les données regroupant les informations sont le coeur même du système d'information. Sans donnée, il n'y a aucune raison de posséder une architecture, d'offrir un service, ou encore d'avoir des choses à protéger. De ce fait, toute donnée doit se voir accorder une attention particulière, et, bien au-delà, doit avoir la possibilité d'être restaurée en cas de de perte du système, de corruption ou de piratage. De nouvelles solutions comme Peerple [1] basées sur la technologie peer-to-peer seront bientôt des alternatives potentielles aux solutions de sauvegarde que nous connaissons actuellement, mais celles-ci manquent encore de maturité afin de pouvoir être mises en place sur une plate-forme de production. Dans une autre optique, on peut envisager l'utilisation de robots de sauvegardes pilotés logiciellement, par l'intermédiaire d'un logiciel comme Amanda [2]. Cependant, pour un usage optimisé, un robot matériel serait nécessaire, ce dernier n'étant pas à notre disposition pour cette maquette, nous avons dû chercher une solution alternative. Celle-ci trouve son issue dans le logiciel *rsync* qui permet la synchronisation locale et distante, avec de nombreuses options ainsi que divers protocoles de transport.

1. Présentation de rsync

Rsync [26] a été créé par Andrew Tridgell et Paul Mackerras. Lors de sa création, entre 1996 et 1999 [27], l'idée principale de ce logiciel est de pouvoir synchroniser deux fichiers sur deux machines

[1] http ://www.peerple.net/ développé par l'INRIA
[2] Advanced Maryland Automatic Network Disk Archiver - http ://www.amanda.org/

distantes l'une de l'autre, en utilisant un lien bas débit. Il est alors nécessaire de préserver la bande passante pour ne transférer que les données utiles. Pour se faire, Tridgell écrit la méthodologie suivante, décrite sous le nom *algorithme de rsync*. Nous avons deux ordinateurs nommés α et β, ayant respectivement un fichier nommé A et B. Ces derniers sont *similaires*, c'est à dire qu'ils ont la même souche, sont initiés du même fichier. Voici l'algorithme :

1. β découpe le fichier B en une série de blocs à taille fixe de S octets non recouvrants. Le dernier bloc peut par conséquent être plus court que S octets.
2. Pour chacun de ces blocs, on calcule deux sommes de vérification : une somme de vérification faible *roulante* et une somme de vérification MD4 forte de 128 bits.
3. β envoie ces sommes de vérification à α.
4. A recherche dans A des blocs de longueur de S octets, cela pour chaque offset et pas seulement pour les multiples de S, qui ont la même somme de vérification faible et forte qu'un des blocs de B. Ceci est réalisable très rapidement en une seule fois en utilisant la propriété de roulement de la somme de contrôle faible.
5. α envoie à β une séquence d'instructions pour construire une copie de A. Chaque instruction est soit une référence à un bloc de B, ou des données littérales. Ces dernières sont seulement envoyées pour les sections de A qui ne trouvent aucune correspondance parmi les blocs de B.

En fonctionnant ainsi, le nombre d'aller-retours de la communication est minimisé à un trajet. *Rsync* est également en mesure de paralléliser ses process, en initialisant un process qui génère et envoie les sommes de contrôle alors que le second reçoit les nouveaux blocs et construit les fichiers. Il en résulte une meilleure utilisation de la bande passante en cas de congestion, puisque *rsync* peut gérer indépendamment les envois et réceptions.

Voici quelques fonctionnalités supplémentaires embarquées dans *rsync* :

- le support de la copie des liens, périphériques, propriétaires, groupes et permissions.
- un comportement similaire à *tar* pour les fonctions *exclude* et *exclude-from*
- un comportement similaire au mode d'exclusion de *CVS* pour ignorer les fichiers tout comme le ferait CVS.
- la possibilité d'utiliser un shell distant, par l'intermédiaire de *ssh* ou *rsh*.
- les privilèges du super utilisateur (*root*) ne sont pas nécessaires pour fonctionner.
- le transfert de plusieurs fichiers simultanément pour minimiser les coûts de latence.
- le support des démons rsync anonymes ou authentifiés (prévu spécialement pour les cas de mirroring).

2. Mise en place technique

Pour nos besoins, nous pouvons être confrontés à deux cas de figure. Le premier correspond à un mirroring, une réplication, des données importantes de configuration ou d'utilisation des serveurs. Pour cela, nous configurons le démon rsync sur le serveur devant être sauvegardé, dans le fichier `rsyncd.conf` :

```
use chroot = no
max connections = 4
hosts allow = 192.168.0.10/32
uid = backup
gid = backup
```

```
[mysql]
        path=/var/lib/mysql


[etc]
        path=/etc
```

Les options utilisées signifient que l'utilisateur et le groupe pouvant intervenir sont `backup`, que seule la machine `192.168.0.10` est autorisée à se connecter, avec une limite de 4 connexions simultanées. Les répertoires cibles sont `/var/lib/mysql` et `/etc`. L'option chroot est invalidée, puisqu'elle obligerait à fonctionner en mode administrateur pour pouvoir changer de racine. On démarre le démon rsync sur le port 873, en écoute des demandes.

Il ne reste plus, sur le serveur destiné à la sauvegarde, qu'à configurer un script dans le fichier de *crontab*, afin que la sauvegarde s'effectue de manière régulière :

```
#!/bin/bash

/usr/bin/rsync -av eof.eu.org::etc /home/save/eof/etc

if [ $? -ne 0 ];then
        mail eof-admin@eof.eu.org -s \
"Problème durant le backup etc de eof" </dev/null
fi

/usr/bin/rsync -av eof.eu.org::mysql /home/save/eof/mysql

if [ $? -ne 0 ];then
        mail eof-admin@eof.eu.org -s \
"Problème durant le backup mysql de eof" </dev/null
fi
```

Dans cet exemple, on appelle le répertoire distant simplement en utilisant le nom défini entre crochets dans la configuration du serveur cible. Le test conditionnel permet de s'assurer qu'en cas d'échec du backup, les administrateurs seront prévenus.

La deuxième option qui s'offre à nous concerne plus particulièrement les répertoires des utilisateurs. Au lieu d'utiliser le démon rsync, nous allons simplement autoriser l'utilisateur à se connecter et nous gérerons les limitations par l'intermédiaire de rssh, voir chapitre ssh.

VI Utilisation de *sudo*

Le système de gestion des droits sous *GNU/Linux* est hérité du modèle Unix, avec un utilisateur appartenant à un ou plusieurs groupes. Chaque utilisateur est en droit de posséder des fichiers en tant que propriétaire. Les droits sur ces fichiers sont divisés en trois parties : les droits du propriétaire, les droits du groupe principal auquel est affilié le propriétaire et les droits du reste des utilisateurs n'entrant pas dans ces deux cas de figure.

Dès lors, chacun de ces groupes se voit possiblement affecter une combinaison des paramètres suivants : - aucun droit, - un droit de lecture du fichier, - un droit d'écriture dans le fichier, - un droit d'exécution du fichier.

Les seules solutions s'offrant à un usager de ces systèmes étaient donc une gestion complexe et difficile des groupes et des droits associés. L'autre solution considérait l'usage de la commande *su*, qui permet de changer d'identifiant utilisateur, à condition de connaître le mot de passe correspondant. Ceci a pour signification que toute commande uniquement et légitimement accessible par l'administrateur système, comme la gestion du pare-feu, ne pouvait être déléguée qu'à la condition d'ouvrir les droits de gestion, ce qui est très dangereux, ou de fournir le mot de passe de l'administrateur, ce qui est inconcevable.

Pour résoudre ce problème, le programme *sudo* a été écrit. Il permet d'exécuter une commande en tant qu'autre utilisateur. L'avantage majeur réside dans le fait que sudo, lors de la demande d'authentification, fait référence au mot de passe de l'utilisateur qui l'invoque et pas à celui dont il souhaite obtenir les droits. Il s'appuie, pour ce faire, sur le fichier de configuration */etc/sudoers* que nous allons regarder en détail.

1. Le fichier de configuration

Pour pouvoir fonctionner, la commande *sudo* s'appuie sur le fichier */etc/sudoers*. Si jamais ce fichier voit la grammaire qui le compose employée incorrectement, la commande ne fonctionnera plus. C'est pourquoi il est important de toujours être certain d'employer la commande *visudo* qui fait la vérification de la syntaxe avant de valider le fichier, ainsi que les droits associés à ce dernier. Dans le cas de l'utilisation d'un éditeur externe, il faudra être sûr de ne pas avoir commis d'incorrection sous peine de rendre la commande inutilisable.

```
# User alias specification
User_Alias      SECURE = jop,mascret
User_Alias      ADMINS = jop,mascret,gusaweb

# Cmnd alias specification
Cmnd_Alias      FW = /sbin/iptables, /bin/netstat

# User privilege specification
root    ALL=(ALL) ALL

SECURE  ALL=NOPASSWD: FW
ADMINS  ALL=NOPASSWD: /bin/su, PASSWD: /usr/bin/tail
```

L'option `User_Alias` permet de définir des groupes d'utilisateurs, `Cmnd_Alias` définit les commandes appartenant ici au groupe `FW` qui permet la gestion du pare-feu et des statistiques réseaux. Les deux dernières lignes définissent les droits attribués à des utilisateurs ou groupes, ainsi que la méthode d'authentification. On peut par exemple voir que les utilisateurs du groupe `ADMINS` peuvent utiliser `/bin/su` sans mot de passe, alors que pour la commande `/usr/bin/tail`, il sera nécessaire de s'authentifier.

2. Une commande sécurisée

La commande *sudo*, lors de sa conception, a été réfléchie de manière à ce que le moins de faiblesses possibles n'apparaissent dans son usage. Cette partie fait donc état des considérations à prendre en compte sur le comportement intrinsèque de cette commande.

sudo permet d'exécuter de manière sécurisée l'exécution de commandes externes. De ce fait, les variables qui gèrent la liaison et le chargement des bibliothèques dynamiques se voient prêter une attention toute particulière. Les variables d'environnement `LD_*` et `_RLD_*` sont supprimées de l'environnement. Le contenu des variables suivantes est également initialisé : `IFS`, `CDPATH`, `ENV`, `BASH_ENV`, `KRB_CONF`, `KRB-CONFDIR`, `KRBTKFILE`, `KRB5_CONFIG`, `LOCALDOMAIN`, `RES_OPTIONS`, `HOSTALIASES`, `NLSPATH`, `PATH_LOCALE`, `TERMINFO`, `TERMINFO_DIRS` et `TERMPATH`. Dans le cas où la variable `TERMCAP` se voit affecter un chemin, celle-ci sera ignorée. Si les variables `LC_*` ou `LANGUAGE` contiennent les caractères `/` ou `%`, elles seront ignorées. Les variables d'environnement qui commencent par un couple de parenthèses sont supprimées, car elles sont interprétées en tant que fonction par l'interpréteur *bash*. Si la commande a été compilée avec le support du *SecurID*, les variables `VAR_ACE`, `USR_ACE` et `DLC_ACE` sont réaffectées.

Pour se prévenir de toute usurpation, une vérification du chemin d'exécution est effectuée. Si le répertoire local est présent, il sera ignoré.

Tout fichier possédant un marqueur de temps positionné dans le futur se verra ignoré. De même, si un fichier possède une date supérieure au ratio $currenttime + 2 * TIMEOUT$, il sera bien évidemment ignoré, mais également signalé dans les fichiers de log.

Certaines commandes, comme les éditeurs de texte (vi, more, joe...), permettent d'invoquer un shell durant leur utilisation. Il faudra veiller à ne pas y laisser accès.

Pour terminer, tout environnement qui supporterait les bibliothèques partagées et qui ne restreindrait pas les chemins de recherche des bibliothèques définies par l'usager pour les programmes *setuid* doit, soit utiliser une option de liaison qui empêche ce comportement, ou lier *sudo* de manière statique.

VII Le noyau Linux

1. Qu'est ce que le noyau ?

Le noyau est une interface entre la couche matérielle et la couche logicielle. Cela signifie que les pilotes permettant de conduire les différents périphériques sont intégrés dans ce noyau. Plus besoin donc de driver externe fourni sur le CD d'un matériel. Non, tout est intégré dans le système central. Ou du moins, presque tout. En effet, bien que Linux soit toujours en développement et malgré les efforts de la communauté du logiciel libre pour maintenir un niveau élevé en terme de compatibilité, certains fournisseurs se refusent encore à donner leurs spécificités. Dans ce cas, seule l'étude d'un driver pour une autre plateforme ou du procédé de communication permettent l'écriture d'un driver pour Linux. On pourrait prendre l'exemple du pilote du modem usb ADSL de chez Alcatel. D'autre part, le noyau intègre une partie logicielle, essentiellement la partie basse du système d'exploitation, comme le processus de lancement du système ou encore la couche réseau et le firewall.

2. Pourquoi compiler un noyau ?

Les distributions actuelles telles que GNU/Debian, Mandriva ou encore RedHat sont prévues pour fonctionner sur une quantité astronomique de machines ayant des configurations matérielles différentes.

Ainsi, beaucoup de pilotes sont chargés en tant que module, afin d'assurer cette compatiblité. De même certaines fonctionnalités logicielles sont absentes, telle que le support de caméra numérique ou encore des fonctions de routage avancées. Il est donc indispensable de pouvoir compiler son noyau afin d'obtenir la version correspondant le mieux à son utilisation, c'est à dire obtenir le noyau le plus adapté à la configuration matérielle du serveur et aux rôles que l'on désire lui attribuer.

a. Ergonomie

Si il existe plusieurs dizaines de type de processeurs, cela est aussi vrai pour les cartes réseaux ou les cartes sons. *Pourquoi emporterions-nous une luge si nous partons en vacances à la mer ?* C'est ce type de question qu'il faut se poser. En effet, s'il nous parait plus adéquat d'emporter un maillot de bain, peut être devrait il en être de même pour notre système d'exploitation. En définissant de manière stricte nos besoins matériels et logiciels, notre noyau n'en sera que plus petit et notre démarrage que plus rapide.

b. Sécurité

Si l'optimisation est en soi une raison suffisante pour recompiler son noyau, l'aspect sécurité vient renforcer ce point de vue. En effet, le noyau possède des capacités de restrictions au niveau utilisateurs qui peuvent, selon l'architecture, être indispensables à une bonne gestion. De même, la suppression d'un certain nombre de fonctions d'un noyau initial nous mettra à l'abri d'attaques en provenance de personnes malveillantes, sur des fonctions dont l'usage ne nous est pas nécessaire. Par exemple, par le passé, un certain nombre de fonctions réseaux se sont retrouvées buggées. Si nous ne les avons pas inclues dans votre noyau de base, il n'y a aucun soucis à se faire quand à un risque potentiel de faille.

3. Patcher son noyau

Il peut arriver que certains programmeurs, par mésentente avec les leaders du développement du noyau ou afin de garder la main sur la manière dont ils développent, n'ont pas pu intégrer le noyau. C'est également le cas de fonctionnalités en cours de développement, qui ne sont là que pour des mesures de test. Ces bouts de code-là peuvent parfois vous être fournis sous forme de patchs. Dans ce cas, il ne vous reste plus qu'à patcher nos sources. Pour ce faire, nous déposons le patch dans le répertoire source de notre noyau puis utilisons la commande suivante :

```
patch -p0 <le_patch
```

Nous n'avons plus qu'à configurer le noyau avec ses nouvelles options maintenant présentes dans notre fichier de configuration. Attention cependant, si l'on relance une fois cette commande, l'opération inverse sera effectuée, c'est à dire que le patch sera enlevé de nos sources.

4. Le patch noyau GRsecurity

GRsecurity est une approche innovante de la sécurité, écrite par Brad Spengler, utilisant une multi-couche de détection, prévention et modèle d'isolement développé sous licence GPL. Il peut être mis en opposition au patch *SElinux*, développé par la NSA. Le fonctionnement de *GrSec* introduit la restriction sur les processus ainsi qu'une étendue des logs. On peut noter les fonctionnalités suivantes :

- un contrôle d'accès basé sur le rôle qui permet de gérer une politique de gestion pour le système sans configuration ;

- durcissement du changement de racine (chroot) ;
- prévention de *chemins* dans `/tmp` ;
- extension de l'audit ;
- prévention de l'exécution de code arbitraire ;
- extension de l'aléatoire dans la pile et les bibliothèques ;
- restriction de la vision des processus à l'utilisateur seul ;
- alertes de sécurité contenant l'adresse IP de la personne qui a déclenché l'alerte.

Ce patch permet donc de sécuriser le système et d'augmenter la verbosité des logs. Nous l'appliquons sur chacun des systèmes qui compose notre architecture.

VIII Intégrité des fichiers du système

Lorsque le système est configuré, que le nouveau noyau est installé, que le chargeur de démarrage est sécurisé, il ne reste plus qu'à utiliser le serveur. Cependant, en l'état, nous sommes dans l'incapacité de garantir l'intégrité des fichiers présents sur notre système. Ainsi, s'il advenait qu'un fichier de configuration système ou qu'un binaire soit modifié, par exemple par un utilisateur malveillant, nous ne pourrions l'apprendre. Pour ce faire, les *Host based Intrusion Detection System* ou *HIDS* ont été inventés. Ils permettent, à partir d'un fichier de signature, de s'assurer qu'aucun fichier n'a été modifié sans que l'administrateur n'en soit averti. L'avantage clairement exprimé est qu'aucune modification sur les fichiers surveillés ne pourra être faite à l'insu du responsable du système. Cependant, les contraintes sont suffisantes pour rendre l'utilisation lourde au quotidien. En effet, la vérification doit s'effectuer quotidiennement et l'analyse humaine est obligatoire pour pouvoir traiter les *faux positifs*. De plus, chaque fois qu'une modification sur un fichier de configuration ou qu'une mise à jour est sollicitée, il faudra recréer l'arbre des sommes de vérification. De plus, il faudra obligatoirement exporter une copie saine de la base lors de sa création, pour s'assurer, en cas d'intrusion, que le fichier local n'a pas été modifié. Nous proposons ici une solution *maison* pour mettre en place notre *HIDS*. D'autres produits comme *tripwire* permettent également un suivi de ce genre.

1. Solution interne

Dans le cas ou l'administrateur souhaite rester maître de sa chaîne de sécurité, il possède suffisamment d'outil d'empreinte et de signature numérique pour pouvoir faire une analyse de bout en bout. Le travail est trivial et sans grande difficulté, il nécessite cependant un certain nombre de manipulations qui, dans la durée, pourront lasser et feront délaisser la surveillance.

```
# Script de création de la base d'empreintes
# JPG - 12/03/2007 - licence GPLv2
#!/bin/bash
# Attention, uniquement le premier niveau des répertoires
REP="/etc /sbin /bin /usr/bin /usr/sbin"

for i in $REP
do
/usr/bin/md5sum $i/* >> /root/hids
done
```

```
/usr/bin/sha512sum /root/hids > hids.full

echo "Signature du script : rentrez votre passphrase"
gpg -s --detach hids.full
```

Dans ce script, nous exprimons les répertoires que nous désirons voir surveiller, puis nous effectuons une empreinte md5 sur chaque fichier contenu dans chacun de ces répertoires que nous envoyons dans le fichier `hids` du répertoire `/root`. Lorsque la création des empreintes est achevée, nous prenons l'empreinte du fichier `hids`. En effet, lors de la vérification, c'est le fichier `hids.full` qui sera d'abord traité. S'il est identique à celui que nous possédons initialement, cela signifie qu'aucun fichier n'a été modifié. Dans le cas contraire, il faudra faire une analyse de `hids` pour savoir sur quoi portent les changements. Pour terminer, nous signons numériquement le fichier `hids.full`, ce qui nous permettra de nous assurer que personne ne sera en mesure de modifier l'intégrité de ce fichier. Il est important de conserver une copie des fichiers `hids` et `hids.full` à l'extérieur du serveur concerné, idéalement sur le poste de l'administrateur ou mieux, sur un cdrom.

Voici le script de vérification quotidienne qui sera lancé et qui avertira l'administrateur en cas de modification :

```
# Script qui vérifie quotidiennement les binaires
# JPG - 12/03/2007 - licence GPLv2
#!/bin/bash
# Attention, uniquement le premier niveau des répertoires
REP="/etc /sbin /bin /usr/bin /usr/sbin"

#Initialisation des fichiers journaliers
cat /dev/null > /root/hids.daily
cat /dev/null > /root/hids.full.daily

for i in $REP
do
/usr/bin/md5sum $i/* >> /root/hids.daily
done

/usr/bin/sha512sum /root/hids.daily > hids.full.daily

va=$(cat /root/hids.full | awk '{print $1}')
vb=$(cat /root/hids.full.daily | awk '{print $1}')

if [ $va != $vb ]
then
/usr/bin/diff -n hids hids.daily > difference
mail -s "modification de fichiers !" admin@eof.eu.org < difference

fi
```

Le script, comme son prédécesseur, reprend l'empreinte de tous les fichiers demandés. Par la suite, une comparaison sur la deuxième somme de contrôle (*SHA 512*) est effectuée. Si celle-ci est modifiée, cela signifie que des différences sont apparues parmi les hashes md5, et donc qu'un fichier a été modifié. Une

différentiation de `hids` et `hids.daily` est alors effectuée et le résultat est envoyé à l'administrateur qui saura immédiatement quels fichiers ont été modifiés.

IX Interrogation distante : *SNMP*

Pour terminer la configuration de la surveillance de nos systèmes, nous allons aborder *SNMP*. *Simple Network Management Protocol* a été créé pour simplifier la vie des administrateurs réseaux, afin de pouvoir interroger leurs systèmes à distance, voire d'interagir. Pour ce faire, des *mibs* ont été associées aux matériels. Elles définissent les options d'action et de surveillance possibles.

De nombreuses informations peuvent être transmises via *snmp*, c'est pourquoi il est important de bien configurer son serveur, voire, lorsque c'est possible, utiliser la version 3 de ce protocole qui ajoute l'usage de la cryptographie dans le protocole.

Si le protocole snmp est très riche et utile, il se peut que les outils de base fournis soient un peu rudes pour une utilisation quotidienne de surveillance. De ce fait, un certain nombre d'outils a été développé. Nous ne verrons pas ici l'utilisation d'outils centralisés comme *Nagios*, *Hobbit* ou encore *Cacti/RRDTools*. Cependant, à partir des résultats annoncés par l'intermédiaire de *snmp*, il est facile de grapher une utilisation du processeur ou de la mémoire, par exemple. Il permet en effet de surveiller l'intégralité d'un parc de machine, quel que soit leur système d'exploitation, en partant de la connectivité en descendant jusqu'au nombre de processus présents sur le système. L'avantage de cette solution est sa capacité à centraliser de manière claire l'état de votre parc à un instant t. Toute solution comme la supervision par mail ou sms peut par la suite être envisagée.

Voici la ligne de configuration à mettre en place dans le démon snmp pour autoriser l'interrogation distante uniquement par notre serveur de supervision, en lecture seule :

```
com2sec readonly 192.168.0.11 eofadmin
```

Sur le serveur de supervision, on pourra alors récupérer les informations nous intéressant, afin de les communiquer aux logiciels adéquats ou de les traiter par nous-mêmes. Voici un petit script qui récupère le taux d'inactivité du processeur et l'état de la mémoire :

```
# Script de récupération des données via SNMP
# JPG  - 11/03/2007 - GPLv2
#!/bin/bash

# Définition des commandes et options
SG="/usr/bin/snmpget"
VS="2c"
COM="eofadmin"
HOST="192.168.0.1"

# CPU idle
CPU=".1.3.6.1.4.1.2021.11.11.0"
# Total memory
MEMTOTAL=".1.3.6.1.4.1.2021.4.5.0"
# Free memory
MEMFREE=".1.3.6.1.4.1.2021.4.6.0"
```

```
$SG -O n -v $VS -c $COM $HOST $CPU $MEMTOTAL $MEMFREE
```

Le résultat retourné par la commande est le suivant :

```
.1.3.6.1.4.1.2021.11.11.0 = INTEGER: 99
.1.3.6.1.4.1.2021.4.5.0 = INTEGER: 515940
.1.3.6.1.4.1.2021.4.6.0 = INTEGER: 74756
```

Chapitre 9

Journalisation

I Conserver les traces

Un système d'information se constitue d'un ensemble de logiciels programmés pour accomplir des tâches répétitives. Celles-ci peuvent être déclenchées par une intervention humaine, ou artificielle, dans le cadre de robots logiciels. Chaque ordre est alors l'initiateur de l'action d'exécution. En réunissant la date d'exécution, la machine issue du système d'information, l'instigateur de la demande, le logiciel impacté et l'ordre passé, nous sommes en mesure de construire un historique d'utilisation [28] du système d'information. Cette collecte d'informations, communément appelée *logs*, est une des rares discipline de l'Informatique à ne posséder un intérêt qu'a posteriori. En effet, ces données sont utiles afin d'identifier un problème (mauvais fonctionnement, mauvaise configuration,etc) au sein du système ou de tracer les agissements d'un utilisateur. Cette dernière mesure est d'ailleurs renforcée par la loi [29] qui oblige un fournisseur d'accès ou un hébergeur à conserver ces données dans le cadre d'investigations judiciaires. Ces traces sont donc une condition sine qua none à la mise en place d'une architecture. Nous pouvons décrire les exigences de ce *service* selon trois problématiques :

- la confidentialité,
- l'intégrité,
- la disponibilité.

La confidentialité impose que les données entreposées ne soient accessibles qu'aux personnes autorisées. Ainsi, la mise à disposition sur Internet serait une violation de la vie privée des utilisateurs et pourrait mener à une action judiciaire. Il est très important de s'assurer de l'accès à ces informations, quitte à ce que ces accès soient eux-mêmes enregistrés.

L'intégrité assure que les données conservées relatent avec exactitude les faits. Lorsqu'ils sont enregistrés, les logs ne doivent alors plus être modifiés. Nous sommes en situation de lecture et d'ajout. Aucune autre action ne doit être autorisée. De même, si le système d'information est impacté par une intrusion, les archives ne devraient pas être détériorées : le cas échéant, il serait impossible de retrouver trace des actions effectuées sur le système.

La disponibilité garantit l'accès aux archives, la consultation des événements passés. Une sauvegarde sous un format logiciel ou physique que nous ne serions pas en mesure d'ouvrir est un exemple de non disponibilité.

Tout système à but de traçabilité doit être cohérent sur l'heure utilisée, il est donc primordial de s'assurer de la synchronisation de l'horloge sur chaque système concerné.

Pour notre étude, nous avons choisi deux types d'archivage. Le premier, local, autorise la consultation des erreurs logicielles en phase d'installation. Il sert également de leurre en cas d'intrusion, le pirate, cherchant en général à effacer ses traces, pourra trouver satisfaction sur les fichiers locaux. Le deuxième, de type client/serveur, permet de conserver sur un serveur protégé les traces des différents serveurs de l'architecture.

II Syslog-ng

Le logiciel Syslog-ng, pour *new generation*, vient en replacement de son ancêtre, syslog. Il a pour but la centralisation et l'archivage des logs, de manière locale ou distante. Alors que son prédécesseur utilisait le couple *facilité* [tableau 9.1]/*sévérité* [tableau 9.2], syslog-ng est capable, en sus, de filtrer les messages en fonction de leur contenu et de les rediriger vers le fichier le plus approprié. La figure 9.1 illustre le schéma de fonctionnement au sein de l'architecture du système d'information. Sur l'hôte client, un logiciel client est installé et reçoit les informations directement des logiciels applicatifs, dans des fichiers (appelés *source*), présents dans le répertoire dédié aux archives logs. Les filtres sont ensuite appliqués et redirigent, selon la configuration, les messages importants vers notre serveur de logs. La deuxième grande avancée du logiciel se trouve ici. En effet, il permet de transmettre des logs à travers les firewall, en utilisant les noms longs des hôtes, ce qui permet de retrouver facilement le client initial, et ce, même si l'on traverse plusieurs pare-feux.

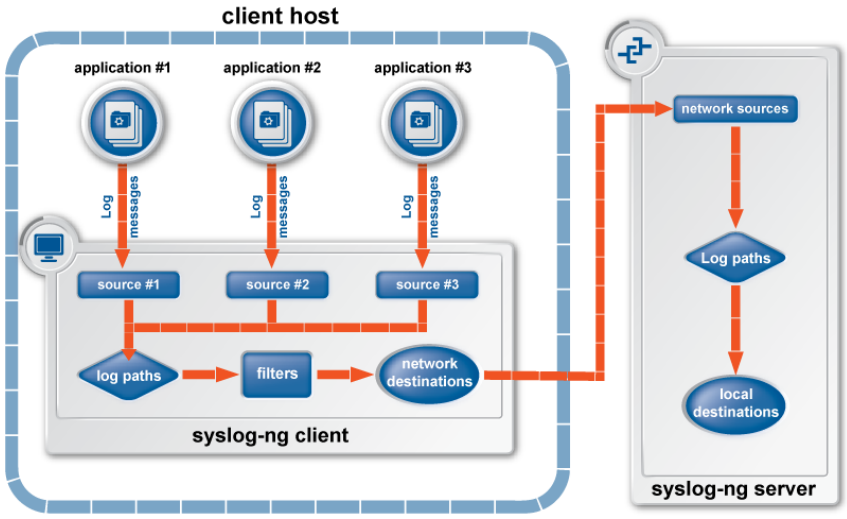


FIG. 9.1 – Fonctionnement de Syslog-ng sur une architecture [annexe F]

La dernière nouveauté réside dans l'amélioration du fichier de configuration pour une lecture plus claire et une maintenance facilitée.

1. Installation de Syslog-NG

Nous utilisons de nouveau les dépôts de la distribution GNU/Debian pour installer notre logiciel :

```
aptitude install syslog-ng
Les NOUVEAUX paquets suivants vont être installés :
  syslog-ng
```

```
Les paquets suivants seront ENLEVÉS :
  klogd sysklogd
```

Il est important de noter que l'ancien service syslog est supprimé. De ce fait, la surveillance du système ne sera effective que lorsque le nouveau service sera correctement configuré. Par conséquent, il est recommandé de faire le choix du système d'archivage avant l'installation, même si un remplacement a posteriori reste envisageable.

2. Configuration de Syslog-NG

Tirant profit de l'expérience de son prédécesseur, mais également des autres types de services, Syslog-ng dispose d'un fichier de configuration segmenté, modulaire, simple, entièrement paramétrable et qui veille et s'intéresse à la sécurité de son système. Il dispose d'une configuration de type objet découpé comme suit :

- périphérique source : méthode de communication utilisée pour recevoir les messages d'archive ;
- source : collection des périphériques sources ;
- périphérique destination : méthode de communication utilisée pour envoyer les messages d'archive ;
- destination : collection des périphériques destinations ;
- filtre : expression pour sélectionner des messages ;
- log : ensemble des sources, filtres et destinations ;
- template : structure définie par l'utilisateur permettant de restructurer les messages d'archive ou les noms de fichiers générés automatiquement ;
- option : paramètres globaux ;

Chaque source doit être identifiée par un nom unique et définir les périphériques qu'elle souhaite supporter :

```
source <identifier> { source-driver(params); source-driver(params); ... };
```

Par exemple, nous configurons un ensemble de sources qui proviennent du fonctionnement interne de syslog-ng, de la queue *log*, du noyau linux, mais également du réseau sur le protocole udp :

```
source s_all {
        internal();
        unix-stream("/dev/log");
        file("/proc/kmsg" log_prefix("kernel: "));
        udp();
};
```

De même, chaque destination doit être définie de manière unique :

```
destination <identifier> {
          destination-driver(params); destination-driver(params); ... };
```

Dans l'exemple ci-dessous, nous déclarons que la destination des archives est un fichier qui prendra pour chemin un certain nombre d'arguments (l'adresse IP ou son nom DNS, l'année, le jour, le mois) et un identifiant unique pour chaque jour (la concaténation de la facilité, de l'année, du mois, du jour).

Ces fichiers et répertoires appartiendront à l'utilisateur *root* avec des permissions bien précises (lecture et écriture autorisées au propriétaire uniquement pour le fichier, agrémentées du droit d'exécution pour le répertoire) et la possibilité de créer le répertoire s'il n'existe pas :

```
  destination hosts {
   file("/var/log/HOSTS/$HOST/$YEAR/$MONTH/$DAY/$FACILITY$YEAR$MONTH$DAY"
   owner(root) group(root) perm(0600) dir_perm(0700) create_dirs(yes));
  };
```

Les filtres autorisent un traitement selon différents critères (facilité, priorité, chaîne de caractères ...) :

```
filter <identifier> { expression; };
```

Par exemple, on récupère tous les messages en provenance du noyau :

```
filter f_kern { facility(kern); };
```

L'objet *log* se positionne en dernier puisqu'il associe les autres objets pour obtenir le comportement final :

```
log {
    source(s1); source(s2); ...
    filter(f1); filter(f2); ...
    destination(d1); destination(d2); ...
    flags(flag1[, flag2...]);
};
```

On retrouve ainsi les sources, filtres et destinations décrits précédemment. Les drapeaux (*flags*) se définissent en quatre possibilités qui peuvent influencer le comportement de syslog [1]. Voici un exemple d'implémentation de l'objet *log* :

```
log {
source(s_all);
filter(f_kern);
destination(hosts);
};
```

Les options sont globales, il est conseillé de les inscrire en début de fichier. Elles se configurent simplement, il suffit d'*appeler* l'une d'entre elle et de positionner ses paramètres :

```
options { option1(params); option2(params); ... };
```

Parmi les options générales, deux méritent une attention particulière. La création de répertoires avec les droits et utilisateurs associés autorise le service à créer les répertoires d'accueil des fichiers s'ils n'existent pas. Dans le cas contraire, le service peut ne pas démarrer. Nous activons cette option :

```
create_dirs(yes);
```

[1] L'usage étant très spécifique, nous vous reportons à la documentation pour d'avantages de détails.

Il est possible lors de la mise en place des archives de lier chaque adresse IP à son identifiant DNS. Cependant, en cas de non réponse du mécanisme DNS en un point de la chaîne, les archives seront bloquées, ce qui provoquera un déni de service. Nous désactivons par conséquent l'usage des dns dans nos archives :

```
use_dns(no);
```

Nous avons pu évoquer deux éléments importants que sont les facilités et les priorités. Les premières, présentées dans le tableau 9.1, regroupent l'ensemble des services ou processus qui sont en mesure de provoquer des remontées d'archives :

TAB. 9.1 – Tableau de correspondance des codes et des facilités

Code	**Facilité**
0	message noyau
1	user-level messages
2	système mail
3	démons système
4	messages de securité/autorisation
5	messages générés en interne par syslogd
6	sous-système imprimante
7	sous-système news
8	sous-système UUCP
9	démon clock
10	messages de sécurité/autorisation
11	démon FTP
12	sous-système NTP
13	log audit
14	log alert
15	démon clock
16-23	facilités utilisées localement (local0-local7)

Les deuxièmes, décrites dans le tableau 9.2, définissent le niveau d'importance, de criticité des messages transférés :

TAB. 9.2 – Tableau de correspondance des codes et des sévérités

Code	**Sévérité**
0	Urgence (Emergency) : le système est inutilisable
1	Alerte (Alert) : une action doit être prise immédiatement
2	Critique (Critical) : conditions critiques
3	Erreur (Error) : conditions d'erreur
4	Avertissement (Warning) : conditions d'avertissement
5	Notification (Notice) : condition normale mais significative
6	Information (Informational) : messages d'information
7	Debugage (Debug) : messages de debuggage

Ces deux paramètres peuvent être directement pris en compte lors de la configuration du logiciel dans les objets *source*, *destination* ou *filtre*.

III Utilisation de swatch

Notre système d'archivage des logs est à présent opérationnel. Il va stocker toutes les informations nécessaires et suffisantes à la surveillance de notre système ou à l'analyse a posteriori de problèmes. Cependant, pour certains événements essentiels, il peut s'avérer nécessaire de mettre en place un *chien de garde*, communément appelé *watchdog*. Toute authentification de l'administrateur système réussie peut ainsi être supervisée, ce qui permet d'être alerté d'une intrusion, ou encore de s'assurer que l'espace disque n'est pas pleinement rempli.

Pour ce qui nous concerne, l'activité humaine, c'est à dire la connexion d'un utilisateur via un shell, est relativement rare sur notre plate-forme. Seuls les administrateurs possèdent les droits nécessaires et suffisants aux actions de configuration des serveurs. Ainsi, nous souhaitons nous assurer que toute élévation de droit ou connexion au système sera reportée aux administrateurs. Pour cela, nous configurons le fichier `/etc/swatchrc` de la manière suivante :

```
# Swatch configuration file for constant monitoring

# Surveillance de sudo
watchfor /USER=root/
        mail=admin-bla@eof.eu.org, subject="Emploi de la commande sudo"
        threshold type=both,count=1,seconds=10

# Surveillance des connexions ssh par mot de passe
watchfor /Accepted keyboard-interactive/
        mail=admin-bla@eof.eu.org, subject="Connexion ssh par mot de passe"
        threshold type=both,count=1,seconds=30

# Surveillance des connexions ssh par clé publique
watchfor /Accepted publickey for/
        mail=admin-bla@eof.eu.org, subject="Connexion ssh par clé publique"
        threshold type=both,count=1,seconds=30
```

Lorsque le logiciel se lance, il consulte le contenu du fichier de configuration. Ici, l'option `watchfor` prend en paramètre une expression rationnelle. L'option `mail` permet de définir les adresses de destinations des courriels d'alerte, ainsi que le sujet de cette dernière. Pour finir, l'option `threshold` permet de mettre en place des limitations à la fois dans le temps et dans la répétition. Nous demandons par exemple, pour l'utilisation de la commande sudo, d'être averti dès le premier usage de la commande, mais pas plus d'une fois toutes les dix secondes. Pour la connexion par ssh, l'attente avant la signalisation d'un nouvel événement est positionnée à 30 secondes.

Pour finir la mise en place de notre outil, il suffit d'inclure la commande de démarrage suivante dans les scripts d'initialisation du serveur :

```
/usr/bin/swatch --daemon --config-file=/etc/swatchrc \
 --tail-file="/var/log/auth.log"
```

La commande se lance dès lors à chaque démarrage du serveur, en utilisant le fichier de configuration situé dans `/etc` et surveille le fichier de log `auth.log`, siège des archives relatives à l'authentification. Nous aurions pu demander à *swatch* de superviser plusieurs fichiers de logs en séparant simplement leur chemin par un espace :

```
/usr/bin/swatch --daemon --config-file=/etc/swatchrc \
--tail-file="/var/log/auth.log /var/log/messages"
```

Notre système de supervision est maintenant au point, l'activité est archivée et des alertes précises nous serons remontées en cas de connexion à notre système.

Chapitre 10

Pare-Feu

Le Pare-Feu est une traduction adéquate de l'expression anglaise *firewall*. Un sens plus intéressant, ou du moins plus intégré à notre quotidien serait *coupe-feu*, tout comme les portes destinées à cet usage dans de nombreux bâtiments. En effet, le pare-feu s'érige en coupure du réseau et permet, selon la configuration mise en place, tantôt d'autoriser la libre circulation d'un flux, tantôt de la refuser ; une autre image moderne s'apparente au *videur* de boîte de nuit : *tu rentres, tu rentres pas, tu rentres, tu rentres pas* ... À l'évidence, c'est donc un filtre, qui peut s'intéresser aux couches deux (*liaison de données*) et trois (*réseau*) du modèle OSI. Au-delà, on ne parle plus de pare-feu, mais de proxy applicatif.

I L'intérêt d'un pare-feu

Tout comme il semble bon de construire une maison avec des fenêtres et des portes, il est intéressant d'insérer un (ou plusieurs) pare-feu dans une architecture informatique. Cela permet de cibler les canaux de communication utiles et souhaités et de stopper net tout le reste. Il autorise également le suivi, au travers des logs, des différentes connexions qui s'établissent entre le monde extérieur et notre infrastructure. Enfin, dans le monde interne, il permet de s'assurer du dialogue normal entre nos équipements.

II Netfilter/Iptables

Dès la version 1.1 de linux, Alan Cox a porté une première version de filtre basé sur *IPFW* de BSD. Il a été amélioré dans la version 2.0, entre autres par Jos Vos, mais ce n'est pas l'unique contributeur. Au milieu de l'année 1998, Rusty Russel et Michael Neuling retravaillèrent le noyau et introduisirent l'outil *ipchains*. Ipchains existe sous la branche linux 2.2. La génération 2.4 du noyau s'est vue dotée d'un outil de la quatrième génération nommé *iptables*.

Ce dernier est actuellement maintenu par l'équipe de Netfilter [1]. Il se décompose en trois tables majeures et cinq chaînes pré-compilées :

- la table *filter* invoquée par défaut, composée des chaînes *INPUT*, *OUTPUT* et *FORWARD* ;
- la table *nat* composée des chaînes *PREROUTING*, *OUTPUT* et *POSTROUTING* ;
- la table *mangle* composée des cinq chaînes *PREROUTING*, *INPUT*, *FORWARD*, *OUTPUT* et *POSTROUTING* ;

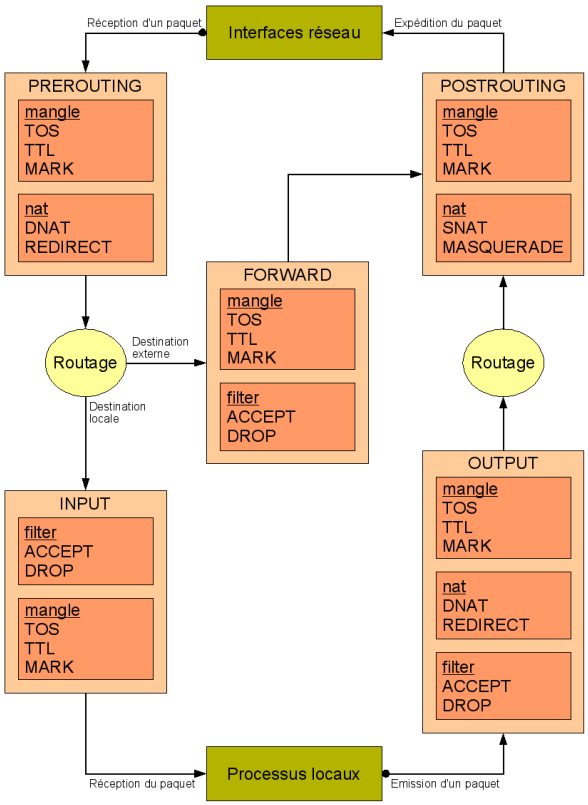


FIG. 10.1 – Fonctionnement de Netfilter

La figure 10.1 présente les différentes interactions possibles entre les tables, les chaînes et le design général, selon la configuration du pare-feu et le chemin que parcourt un paquet au sein de ce dernier.

Une table est absente de ce schéma car elle traite l'exception. Il s'agit de la table *raw* qui se positionne en priorité devant toutes les autres. Elle permet d'écarter le suivi de connexion sur certains paquets en corrélation avec la cible *NOTRACK*.

III Le suivi de connexion

Les premières implémentations de firewall étaient des filtres simples qui se contentaient de définir si un service devait pouvoir être joint ou non. Pour ce faire, le couple *IP/port* était observé, et, s'il s'avérait valide, la requête était alors acheminée. La procédure énoncée est peu sûre et engendre un problème connu et identifié. Il s'agit en fait d'une usurpation du port source. Pour accéder à une application *firewallée*, l'utilisateur cherche un port ouvert et le place en source dans son paquet. Si l'on prend l'exemple d'un serveur web que l'on souhaite atteindre, mais qui est protégé, alors que le service de données de FTP (port 20) est ouvert, sur un firewall simple, on pourra outrepasser la sécurité avec une simple commande :

```
netcat -v -p 20 192.168.0.3 80
```

On écoute en mode verbeux sur le port local 20 vers la machine distante `192.168.0.3` sur le port 80. On peut également transformer nos paquets à partir de Netfilter/IPtables lui-même :

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

```
iptables -t nat -A POSTROUTING -p TCP -m tcp -s 192.168.0.2 \
--dport 80 -j SNAT --to-source 192.168.0.2:20
```

Nous utilisons le module de `POSTROUTING`, c'est à dire que tous les paquets sortants, ayant pour adresse source `192.168.0.2` et pour destination le port 80 seront modifiés pour avoir le port source d'émission 20.

On note que cette manipulation n'a pas de rapport direct avec le suivi de connexion, puisque ce dernier maintient des tables sur l'établissement des connexions, le sens de celles-ci, les ports mis en jeu, etc. Le port 20 est, comme nous le disions, le service data du protocole FTP. Ce dernier ne doit être atteint que lorsqu'un échange valide est opéré par l'intermédiaire du canal de contrôle (port 21 du service FTP). Ainsi, un pare-feu correctement configuré autorise uniquement les connexions valides (nouvelle connexion, ou trafic relatif à cette connexion), avec les ports sources et destination bien positionnés.

Il existe cinq états en relation avec le suivi de connexion et la table qui y est liée :
- **New** : premier paquet de l'établissement d'une connexion qui permet l'inscription dans la table de *contrack*,
- **Established** : résultat de l'observation d'un trafic bidirectionnel, en général suite à un jeu question/réponse (*three-way-handshake* pour l'exemple),
- **Related** : lié à l'état précédent, une nouvelle connexion est effectuée. Ce cas est usité dans les protocoles tels que IRC ou FTP (qui nécessitent eux-mêmes des modules spécifiques),
- **Invalid** : cet état correspond à un paquet qui n'aurait aucun des états précédents. Il faut en règle général le jeter,
- **Untracked** : la connexion spécifiée ne doit pas être suivie par la table.

Ces états peuvent être utilisés dans les règles du pare-feu par l'intermédiaire de l'appel `state`, comme l'illustre l'exemple suivant, où toutes les connexions valides sont acceptées :

```
IPTABLES -A INPUT -m state \
--state NEW, ESTABLISHED, RELATED \
-J ACCEPT
```

IV Les traces

Au-delà de son rôle de filtre, la deuxième fonction la plus importante consiste à la surveillance des paquets traversant le firewall. Ceci permet à la fois de déterminer si les règles sont correctement écrites, mais également si un utilisateur rencontre réellement une règle ou non. Au final, les traces permettent le deboggage de la politique de sécurité implémentée sur le pare-feu, mais également le suivi de l'activité quotidienne de ce dernier. Il existe deux cibles codées dans *Netfilter* : *Log* et *Ulog*.

La première travaille directement en mode noyau et envoie toutes les traces au démon *Syslog*. Par la suite, Ulog qui fonctionne en espace utilisateur est venu renforcer cette gestion des traces, avec une finesse d'option supérieure, mais également la possibilité de se connecter à une base de données, pour un meilleur traitement postérieur des archives.

Ces deux cibles sont non stoppantes, c'est à dire que tout paquet qui traverse une règle configurée avec l'une d'elles continuera son parcours des règles du pare-feu. Si l'on souhaite un traitement du paquet, il est nécessaire de créer une seconde règle avec un traitement spécifique (acceptation du paquet ou blocage).

Il peut être utile, lors de l'établissement de la politique de règles, de ne mettre en cible que des politiques d'archive, afin de s'assurer que toutes les règles décrites sont correctes, avant de les écrire avec une cible finale.

Voici un exemple d'archive de tentative de connexion SSH :

```
iptables -A INPUT -p TCP \
--dport 22 -j ULOG \
--ulog-prefix "SSH connection attempt: "
```

V Une politique bien définie

Il existe deux manières de gérer une politique de sécurité d'un pare-feu. Soit on autorise tout le trafic et on rejette ce que l'on ne souhaite pas accueillir sur son réseau, soit on arrête tout le trafic et on autorise uniquement celui qui semble correspondre à des demandes légitimes. Dans notre cas, nous adopterons la deuxième méthode qui nous semble plus adéquate en terme de filtrage. Cette politique s'inscrit directement dans les chaînes majeures :

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUPTUT DROP
```

Ainsi, il est nécessaire de référencer uniquement les flux qui ont lieu d'être au sein de notre réseau, ainsi que quelques mesures supplémentaires de précaution. Par exemple, aucune des adresses internes de notre parc ne doit venir de l'extérieur, ni pouvoir être jointes directement. C'est ce qu'on appelle l'*anti-spoofing*, ou la gestion de l'usurpation des adresses IP. Nos serveurs étant sur un même sous-réseau, les paquets ne passeront pas par le pare-feu, nous pouvons donc tout simplement jeter tout paquet qui aurait les mêmes adresses. Il est également possible d'affiner les règles selon l'interface sur laquelle arrivent ou sortent les paquets :

```
iptables -A FORWARD -s 192.168.0.0/16 -j DROP
iptables -A FORWARD -s 10.0.0.0/8 -j DROP
iptables -A FORWARD -s 172.16.0.0/11 -j DROP
```

De même, le port source d'un paquet se doit d'être supérieur à 1023 et peut aller jusqu'à 65535 (1024 :65535) ; de fait, tout paquet dont le port source serait inférieur à 1024 peut être supprimé avec comme hypothèse la détection d'une attaque.

Pour finir, comme nous l'avons vu dans la section précédente, le pare-feu doit intégrer la gestion des états dans ses règles.

a. Pare-feu global

Un pare-feu global se positionne en entrée de site. Il gère de manière générale les flux et ne s'occupe pas des échanges internes issus du réseau, sauf pour dialoguer entre les différentes DMZ. C'est lui qui est mis en frontal avec Internet, il est donc primordial de surveiller son activité. Le matériel est souvent redondé pour pallier à toute panne.

b. Pare-feu par hôte

À l'instar du pare-feu global, chaque serveur est doté d'un pare-feu *personnel* qui permet une gestion très fine des flux autorisés et filtrés. Ainsi, la politique se portera davantage sur les échanges inter-réseau, le trafic externe étant déjà surveillé. La mise en application de ces deux politiques permet une granularité plus grande. C'est ce que l'on appelle la sécurité par couche (ou en profondeur).

VI Une gestion par Objet

Plutôt que d'écrire les règles directement à travers l'interface IPTables, il est plus intéressant de réfléchir à la gestion de nos règles, des serveurs et des services qui vont être impactés, le sens des flux, etc. Nous proposons ici un schéma de nomenclature qui sert à la fois de fichier de définition de notre politique, mais également de documentation. Cette nomenclature se définit par objet, objet que nous pourrons considérer comme des briques de *lego*. Les plus petites briques s'assemblent pour former des briques globales plus conséquentes. La définition de cette granularité autorise une manipulation simple de la formation des règles. Nous proposons un schéma qui reprend l'ensemble des types d'objet et qui permet une documentation de haut niveau, lisible par tous.

Hôtes : définition de l'hôte selon un nom unique préfixé de la racine **h-** (pour host), de son adresse IP et du ou des groupes auxquels il appartient.

Identifiant	IP	Groupe(s)
h-www	192.168.0.1	g-www

Groupes d'hôtes : définition d'un groupe selon un nom unique préfixé de la racine **g-** (pour group), des hôtes qui le composent.

Identifiant	Hôtes
g-www	h-www h-www1 h-www2

Réseau : définition d'un réseau selon un nom unique préfixé de la racine **n-** (pour network) et suffixé par le CIDR, de son adresse de réseau, son masque de sous réseau et du ou des groupes auxquels il appartient.

Identifiant	IP	Netmask	Groupe(s)
n-192.168.0.0-24	192.168.0.0	255.255.255.0	gn-1918
n-172.16.0.0-11	172.16.0.0	255.224.0.0	gn-1918
n-10.0.0.0-8	10.0.0.0	255.0.0.0	gn-1918

Groupes de réseaux : définition d'un groupe selon un nom unique préfixé de la racine **gn-** (pour group of network), des réseaux qui le composent.

Services : définition d'un service préfixé par **s-** (pour service), du type de protocole (TCP/UDP), du port et suffixé par le nom du service si celui-ci est connu. On retrouve également le protocole et le port dans le tableau et le ou les groupe auxquels il appartient.

Identifiant	Réseaux
gn-1918	n-192.168.0.0-24 n-172.16.0.0-11 n-10.0.0.0-8

Identifiant	Protocole	Port(s) source(s)	Port destination	Groupe(s)
s-TCP-80-HTTP	TCP	1024 :65535	80	gs-www
s-TCP-443-HTTP	TCP	1024 :65535	443	gs-www

Groupes de services : définition d'un groupe selon un nom unique préfixé de la racine **gs-** (pour group of services), des services qui le composent.

Identifiant	Services
gs-www	s-TCP-80-HTTP s-TCP-443-HTTPS

NAT : Le Network Adress Translation est un mécanisme qui permet de traduire une adresse par une autre adresse, en général de l'adressage publique vers l'adressage privé et inversement. Il a été mis en place suite à la pénurie d'adresse IPv4 (voir chapitre IPv6 I). Dans notre matrice, il prend la forme suivante :

Adresse	NAT
86.86.84.115	192.168.0.3
86.86.84.116	192.168.0.8

Règles : La règle incorpore tous les objets que nous avons pu décrire précédemment, exception faite pour le NAT qui peut être considéré comme une source ou une destination. De nouveaux champs font également leur apparition. Les zones définissent si le flux provient de l'intérieur du réseau (trust) ou de son extérieur (untrust), la cible renvoie vers une chaîne ou une action (Accept ou Drop), les logs peuvent être activés ou non, ce qui permettra d'avoir une vision claire de ce qui traverse le firewall, et le commentaire permet, à toute fin utile, de savoir à quoi correspond la règle.

Zone-src	Source	Zone-dst	Destination	Service	Cible	Logs	Commentaire
Untrust	Any	Trust	g-www	gs-www	Accept	Logging	Accès aux serveurs web
Untrust	Any	Trust	h-ssh	s-TCP-22-SSH	Accept	Logging	Accès à la machine d'administration

VII La mise en place de nos pare-feux

Tout comme nous avons pu le voir, *iptables* peut à la fois servir en entrée de réseau, mais également en entrée de machine. Ainsi, en positionnant une sécurité de filtrage à deux niveaux, on augmente la *sécurité en profondeur* de notre site.

1. Filtrage en entrée de site

Nous pouvons à présent configurer notre pare-feu d'entrée de site, directement relié à Internet. Il ne s'occupera que des flux entrants (*Untrust* vers *Trust*), avec un suivi de connexion. Pour tout ce qui concerne les mises à jour des systèmes, une règle spécifique sera mise en place temporairement chaque fois que nécessaire. Pour s'assurer qu'il n'existe aucune possibilité de connexion externe en IPv6, nous configurons *ip6tables*, le pendant d'*iptables* pour IPv6, afin rejeter tout paquet qui arriverait sur ses interfaces. Pour cela, nous utilisons les commandes suivantes :

```
ip6tables -P INPUT DROP
ip6tables -P OUTPUT DROP
ip6tables -P FORWARD DROP
```

Il ne reste plus qu'à dresser le tableau des flux qui seront acceptés par notre pare-feu général :

TAB. 10.1 – Règles d'accès au site

Id	Source	Destination	Service	Cible	Logs	Commentaire
1	gn-antispoof	Any	Any	DROP	Logging	Rejet des réseaux 1918
2	Any	g-dns	s-TCP-53-DNS s-UDP-53-DNS	ACCEPT	Logging	Interrogation DNS
3	Any	rwww	s-TCP-80-HTTP s-TCP-443-HTTPS	ACCEPT	Logging	Requêtes web
4	Any	hssh	s-TCP-22-SSH	ACCEPT	Logging	Accès Honeypot
5	Any	vssh	s-TCP-22-SSH	ACCEPT	Logging	Accès d'administration console
6	Any	pki	s-TCP-3333	ACCEPT	Logging	Administration PKI
7	Any	Any	Any	DROP	Logging	Politique par défaut

La règle (1) permet de s'assurer qu'aucune usurpation de l'adresse IP de nos machines n'est possible depuis l'extérieur. Les règles (2) et (3) autorisent l'accès aux services DNS et au répartiteur de charge web. On notera au passage que le protocole TCP est également autorisé pour le service DNS ; cela fait suite à la possibilité qu'une réponse DNS soit plus large que la longueur totale d'un datagramme DNS, ce qui le fait automatiquement passer du mode UDP au mode TCP. Les accès (4) et (5)

sont séparés, bien que filtrant le même protocole. Cette règle permet d'obtenir une ligne de filtrage spécifique pour le pot de miel, et donc de l'adapter en conséquence selon les événements. La ligne (6) gère l'accès au serveur de l'infrastructure de gestion de clé publique. La règle finale définit la politique que nous retrouverons en entête des chaînes génériques.

2. filtrage par machine

Le travail matriciel est à reproduire pour chaque machine qui se voit dotée d'un pare-feu. On y retrouvera bien sûr les règles de non usurpation, ainsi que la politique générale. La spécificité se situe plus particulièrement au niveau de l'accès SSH, les flux Syslog ou la réplication inter-serveur.

VIII Prévention d'intrusion

Le domaine de la détection d'intrusion a évolué vers celui de la prévention d'intrusion. Cette deuxième approche a une forte connotation commerciale, mais elle dénote aussi le changement de position d'une situation a posteriori vers une situation a priori. On ne constate plus pour réagir, on travaille en amont. Le logiciel *Portsentry* est un analyseur de scans (un scan permet au final de son analyse de savoir quel type de service est présent sur un serveur et sur quel port il écoute). Il se place devant un pare-feu et se met en position d'écoute sur un certain nombre de ports. Si un attaquant interroge le serveur port par port, le logiciel archive son IP et communique celle-ci au pare-feu qui bloque la connexion selon une règle définie dans la configuration :

```
KILL_ROUTE="/sbin/iptables -I INPUT -s $TARGET -j DROP"
```

Ce procédé permet de mettre en liste noire toute adresse IP qui tente une suite de connexions sur des ports désactivés, trafic qui n'a pas de sens légitime. Ainsi, de nombreux curieux seront bloqués avant même de pouvoir tenter une attaque sur le serveur.

Cette configuration crée cependant un risque de déni de service. Si un attaquant relève la présence du logiciel, il peut forger un ensemble fini de requêtes qui comporteront toutes les adresses IPv4 disponibles. En activant le mécanisme de défense de Portsentry, chaque IP sera bannie tour à tour. Ainsi on ne pourra plus accéder à la machine, coupée du monde, que par l'intermédiaire d'une console physique. Il est important de prendre en compte cette faiblesse lors de l'utilisation de l'outil. Pour cela, quelques règles simples suffisent :

- positionner une règle du pare-feu qui garantisse toujours l'accès des administrateurs par des IP fixes,
- configurer un *watchdog* qui prévienne l'administrateur si ce type de situation se présente et stoppe le process de bannissement,
- faire un nettoyage régulier des règles, de manière hebdomadaire par exemple. Cette tâche peut être automatisée par un script en *crontab*.

Ce système prévient la majorité des attaques, mais est inefficace contre une attaque ciblée, ce n'est donc pas une réponse absolue, mais un pas de plus dans la sécurisation de la plate-forme.

Chapitre 11

Infrastructure de clé publique

I Rappels sur la cryptographie

1. La cryptographie

La cryptographie est un procédé qui permet de rendre un message inintelligible, de protéger des données. Ainsi, tout message ou donnée que l'on souhaite protéger d'un oeil indiscret peut se voir modifier par un algorithme cryptographique. La figure 11.1 nous illustre cette transformation. Dans la littérature consacrée, nous retrouvons différents termes et usages : cryptage, chiffrement, cryptanalyse, signature... Nous allons illustrer ces différents termes dans les sections suivantes.

FIG. 11.1 – Principe de la cryptographie

2. La cryptanalyse

La cryptanalyse s'intéresse au décodage des messages chiffrés ou aux biais présents dans les algorithmes, ou leur implémentation, qui permettent de retrouver le message initial, comme le montre la figure 11.2.

À partir d'un texte initialement chiffré, on cherche à retrouver le message en clair. Il est possible de rapprocher ce procédé du *reverse engineering* ou *rétro-conception*, à la différence près que le langage machine peut hypothétiquement être remplacé par des masques ou des répétitions. C'est cette étude qui est faite sur le chiffrement de Vigénère, ou plus récemment le cassage du code des machines Enigma. Cette science requiert des moyens mathématiques et informatiques importants, s'appuyant généralement sur de nombreux résultats mathématiques.

FIG. 11.2 – Principe de la cryptanalyse

3. Le chiffrement

Si la genèse de la cryptographie proposait de l'*obfuscation*, en utilisant de la permutation de lettres, par exemple illustré par le *rot13* : rotation de treize lettres de l'alphabet pour chiffrer le message, puis rotation dans le sens inverse pour retrouver le sens initial ; ce n'est très vite plus le cas avec l'emploi d'algorithmes et de jeux mathématiques faisant apparaître la notion de clé (voir figure 11.3). Celle-ci peut être soit confidentielle, soit connue des parties en présence, selon le type de chiffrement. Cette clé, tout comme celle d'une serrure, est unique et permet de sceller les données. C'est à dire que toute personne ayant connaissance de l'algorithme ne sera pas en mesure de déchiffrer un message s'il ne possède pas la clé permettant d'ouvrir cette *serrure*.

FIG. 11.3 – Chiffrement à l'aide d'une clé secrète

Le prédicat initial veut que pour une clé donnée, un seul message chiffré existe. Si pour deux clés, il existe un même message chiffré, on parle alors de collision : le message peut être déchiffré à l'aide de l'une ou de l'autre clé. Un algorithme présentant des collisions n'est pas un bon algorithme de chiffrement.

4. Le déchiffrement

À l'inverse, le déchiffrement est l'opération qui, à partir d'un message chiffré, d'une clé et d'un algorithme spécifique, permet d'obtenir le message initial (voir figure 11.4).

La clé est le vecteur d'aléatoire le plus sûr. Il faudrait essayer toutes les suites de caractères hypothétiques et les confronter à l'algorithme et aux données chiffrées pour espérer revenir au message initial. Ce type d'attaque, nommé *attaque par dictionnaire*, est très efficace dans le cas de clé courte et usuelle. Il est donc conseillé d'employer des clés longues, c'est à dire supérieure à treize caractères, et dont le sens ne peut être connu de personnes (caractères aléatoires, initiales de chaque mot d'une page de livre…).

FIG. 11.4 – Déchiffrement à l'aide d'une clé secrète

5. Condensat

Le condensat, également appelé *hash*, *empreinte*, voire *cryptage* lorsque l'on parle de son algorithme est un procédé qui à une donnée associe une et une seule chaîne de caractères hexadécimale fixe. Cette empreinte est alors garante de l'intégrité de la donnée, pourvu que cette dernière, soumise au même algorithme de condensat, fournisse le même résultat. L'exemple suivant nous propose la génération d'une empreinte dite *MD5*, selon l'algorithme du même nom, correspondante à un fichier image spécifique :

```
md5sum graph02.png
21147474215ae2aba8adbf626f7d65a5  graph02.png
```

La même image soumise à l'algorithme *SHA1* fournira une empreinte différente :

```
sha1sum graph02.png
8bf04bd729610a3b20d390488f327c07cfbed53d  graph02.png
```

Ce procédé est utilisé lors de transferts de fichiers, afin de s'assurer que le fichier n'a pas été modifié dans son intégrité. Pour ce faire, on propose le condensat sur le serveur hôte et on le vérifie sur le serveur destination. Si les deux hashés sont identiques, le transfert s'est effectué correctement.

6. Le chiffrement symétrique

Le chiffrement symétrique est un algorithme qui prend une clé unique pour le chiffrement et le déchiffrement du message. Cette clé doit donc être connue par toutes les personnes qui devront accéder au message chiffré. Plus le nombre de personnes ayant connaissance de la clé est important, moins la sécurité de la confidentialité peut être garantie. Ces algorithmes sont choisis car ils sont simples à mettre en place, même électroniquement. Il suffit par exemple de bascules `AND`, `OR`, `XOR` et leur inverse pour créer un circuit en mesure de chiffrer et déchiffrer à la volée. Voici quelques-uns des algorithmes les plus connus :

- DES
- AES
- Serpent
- Twofish
- Blowfish
- Mars

7. Le chiffrement asymétrique

Le chiffrement asymétrique s'appuie sur une relation mathématique forte, où les nombres premiers ont une importance majeure, et qui associe un couple clé privée/clé publique. Alors que la clé privée est la détentrice du secret, la clé publique peut être communiquée à l'extérieur. Voici les algorithmes asymétriques les plus connus :

- RSA, le plus utilisé d'entre eux
- Cryptosystème de ElGamal
- Cryptosystème de Merkle-Hellman

II Mise en application de la cryptographie

Il existe de nombreuses applications que nous utilisons tous les jours, issues du domaine cryptographique :

- les cartes à puces (cartes bancaires, d'achat, etc),
- le *s* de tous les protocoles Internet dit sécurisés (httpS, ftpS, popS, imapS...),
- le paiement des impôts par voie électronique,
- ...

Nous allons illustrer rapidement un usage courant de la cryptographie au sein de l'administration de la plate-forme. Les administrateurs utilisent le logiciel gpg [1] pour authentifier ou chiffrer les échanges de courrier électronique en cas d'information confidentielle.

1. Signature

La signature consiste à la réalisation d'un condensat que l'on authentifie avec notre clé privée. La clé publique permettra à tout un chacun de s'assurer que l'information provient bien du possesseur de la clé privée, mais également que l'information n'a pas été manipulée. Dans le cas d'un échec de la vérification, on peut considérer, au choix, que le message a été manipulé, ou que l'expéditeur est usurpé.

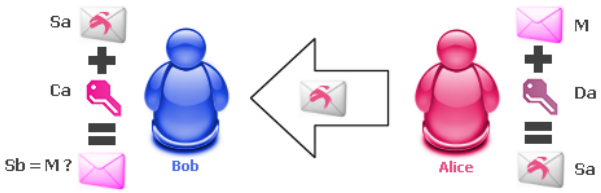


FIG. 11.5 – Processus de signature et authentification d'un courrier électronique

Tout courriel utilisant ce procédé peut alors servir de preuve auprès de la justice ou de l'administration [2].

Un exemple de mail signé est disponible en annexe F.

[1] http://www.gnupg.org

[2] LOI no 2000-230 du 13 mars 2000 portant adaptation du droit de la preuve aux technologies de l'information et relative à la signature électronique. NOR : JUSX9900020L

2. Chiffrement

Le chiffrement permet de conserver la confidentialité des échanges. Ainsi, tout mot de passe ou information peut circuler par mail sans avoir la crainte d'être épié sur un réseau public ou serveur de courriel. Le procédé est un peu inverse à celui de la signature. En effet, pour chiffrer un courriel, il est nécessaire de connaître la clé publique du destinataire. À partir de celle-ci, on peut chiffrer le message. Dès lors, seul le possesseur de la clé privée et de la passphrase associée sera en mesure de déchiffrer les données et de restituer son état initial à l'information.

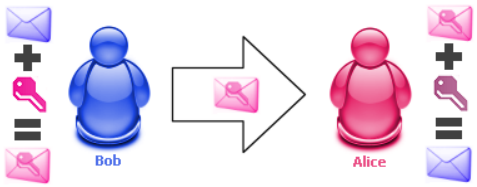


FIG. 11.6 – Processus de chiffrement et déchiffrement d'un courrier électronique

Le chiffrement ne garantit pas l'authenticité du message. Il faut, de fait, coupler la signature au chiffrement pour obtenir une authentification et une confidentialité simultanées. De la même façon qu'énoncé précédemment, md5 ne garantit que ce qu'il garantit, à savoir que le transfert s'est bien passé, mais pas qu'on a téléchargé le bon fichier. Celui-ci a pu être corrompu et mis sur un serveur. D'où l'apparition de paquets désormais signés.

L'emploi de la technologie OpenPGP est une des solutions proposées pour la signature des courriels et des paquets binaires. L'autre est d'utiliser la technologie *S/MIME*, à base de certificats. Ces mêmes certificats sont également utilisés et gérés par les Infrastructures de Gestion de Clés (IGC) que nous allons maintenant aborder.

III Description d'une Infrastructure de Gestion de Clés publiques

1. Importance du rôle de l'IGC

Les IGC (PKI en anglais, pour Public Key Infrastructure) ont été créées afin de pouvoir gérer les contraintes légales, sécuritaires et administratives qui incombent à la gestion des certificats.

Un certificat électronique (ou numérique) est la représentation dématérialisée d'une personne physique ou morale. L'Autorité de Certification (AC), qui délivre les certificats, fait foi de tiers de confiance ou encore de notaire et atteste du lien entre l'identité réelle et son pendant numérique. La norme la plus utilisée pour la création des certificats numérique est le X.509.

Pour résumer la situation d'une IGC, la figure 11.7 illustre la place et le rôle du notaire : il détient les clés privées et clés publiques des utilisateurs. Il met à disposition ces dernières afin que quiconque puisse vérifier l'authenticité d'un document en provenance d'un émetteur ou chiffrer pour un destinataire. C'est un rôle central majeur.

Les rôles sont étroitement liés aux composants de l'IGC. Ils se définissent selon la liste suivante :
- enregistrement des utilisateurs,
- génération de certificats,
- révocation de certificats,
- publication des certificats valides et révoqués,

FIG. 11.7 – Représentation d'une IGC

- identification et authentification des utilisateurs,
- archivage des certificats.

2. Composants

Les composants d'une IGC sont standardisés par un groupe de travail de l'IETF [3] qui a en charge la rédaction des *rfc* de l'art. Ainsi, nous trouvons quatre entités obligatoires, définies dans la rfc 4210 [30] :

- L'Autorité de Certification (AC ou CA) a pour mission de signer les demandes de certificat (CSR : Certificate Signing Request) et de signer les listes de révocation (CRL : Certificate Revocation List), cette autorité est la plus importante.
- L'Autorité d'Enregistrement (AE ou RA) génère les certificats et effectue les vérifications d'usage sur l'identité de l'utilisateur final (les certificats numériques sont nominatifs et uniques pour l'ensemble de l'IGC).
- L'Autorité de Dépôt (Repository) stocke les certificats numériques ainsi que les listes de révocation (CRL).
- L'Entité Finale (EE : End Entity) est optionnelle, cette entité est en charge de l'enregistrement des nouveaux utilisateurs finaux. Les utilisateurs des certificats (machines, personnes physiques et morales) font partie de l'entité finale.

En complément, on pourra ajouter l'autorité de séquestre :

L'Autorité de séquestre (Key Escrow) a un rôle particulier ; en effet lorsqu'on génère des certificats de chiffrement, on a l'obligation légale de fournir aux autorités un moyen de déchiffrer les données chiffrées par un utilisateur de l'IGC. C'est là qu'intervient le Séquestre, cette entité a pour mission de stocker de façon sécurisée les clés de chiffrement qui ont été générées par la PKI, afin de pouvoir les restaurer le cas échéant.

IV Les certificats numériques

Nous venons de parcourir les différentes autorités d'une IGC, il reste maintenant à découvrir les données manipulées par cette dernière, à savoir les certificats.

[3] http://www.ietf.org/html.charters/pkix-charter.htm

1. Définition

Un certificat est un objet qui contient des informations cryptographiques, mais également compréhensibles à la lecture humaine. Il se décompose en un certain nombre de champs obligatoires et optionnels, comme décrit dans la rfc 2459 [31].

Voici comment se compose un certificat de type X.509 :

- Version : est un entier qui propose pour l'instant trois possibilités : v1(0), v2(1), v3(2)
  - Serial Number : est un entier
  - Algorithm ID : définit l'algorithme employé et ses paramètres
  - Issuer ou Issue : est relatif à l'entité dont est issu le certificat. Il doit obligatoirement être complété et être présent sous forme d'un *Distinguished Name*.
  - Validity : définit la durée de vie du certificat
    - Not Before : date avant laquelle le certificat n'a aucune valeur
    - Not After : date après laquelle le certificat n'a aucune valeur
  - Subject : identification de l'entité liée à la clé publique du certificat
  - Subject Public Key Info : transport de la clé publique et identification de l'algorithme
- Certificate Signature Algorithm : algorithme utilisé pour signer le certificat
- Certificate Signature : signature du certificat

Vous trouverez un exemple de certificat en annexe G.

2. Processus de délivrance d'un certificat

Un certificat numérique naît dès lors qu'une demande de certificat a abouti.

Une demande de certificat est un fichier numérique (appelé soit par son format, PKCS#10, soit par son équivalent fonctionnel, CSR pour Certificate Signing Request) qui est soumis à une Autorité d'Enregistrement par un utilisateur final ou par un administrateur le représentant.

Cette demande de certificat est examinée par un Opérateur d'Autorité d'Enregistrement. Cette position est une responsabilité clé : c'est lui qui doit juger de la légitimité de la demande de l'utilisateur et accorder, ou non, la confiance de l'organisation. Pour cela, l'Opérateur doit suivre une série de procédures, plus ou moins complètes, consignées dans deux documents de référence qui vont de pair avec la création d'une IGC, et qui sont la Politique de Certification (PC) et la Déclaration des Pratiques de Certification (DPC). Ces documents peuvent exiger, en fonction des enjeux de la certification, des vérifications plus ou moins poussées : rencontre en face-à-face, validation hiérarchique, etc. L'objectif de l'Opérateur d'AE est d'assurer que les informations fournies par l'utilisateur sont exactes et que ce dernier est bien autorisé à solliciter la création d'un certificat.

Une fois son opinion formée, l'Opérateur de l'AE valide la demande ou la rejette. S'il la valide, la demande de certificat est alors adressée à l'Autorité de Certification (AC). L'AC vérifie que la demande a bien été validée par un Opérateur d'AE digne de confiance et, si c'est le cas, signe la CSR. Ce CSR devient alors un certificat.

Le certificat, qui ne contient aucune information confidentielle, peut par exemple être publié dans un annuaire d'entreprise : c'est la tâche du Module de Publication, souvent proche de l'AC.

3. Fin de vie

Il existe deux scénarios de fin de vie d'un certificat :

– le certificat expire, c'est à dire qu'il arrive au terme légal de sa vie, date après laquelle il ne garantit plus les informations certifiées jusqu'alors,
– le certificat est révoqué et n'est donc plus considéré comme valide.

Le premier cas, le plus courant, est le comportement normal attribué à tout certificat. Il *naît*, *vit* et *meurt*. Cela garantit le renouvellement et le suivi des installations. Le deuxième cas intervient lors d'une compromission du certificat, d'un vol ou d'une perte. L'autorité de certification révoque alors le certificat et ce dernier ne sera plus considéré comme valide, ce qui interdirait par exemple l'authentification sur un site dont le certificat client aurait été révoqué.

V Mise en place de Newpki

Les logiciels de gestion d'infrastructure de clés sont assez nombreux dans l'univers du logiciel libre, mais peu ont atteint une maturité satisfaisante, entendre par là une stabilité et une richesse de contenu, ainsi qu'une documentation capable d'accompagner l'utilisateur final de cette activité. Après avoir étudié un certain nombre d'IGC orienté sur un développement web (rooster, phpki ...), nous sommes finalement revenu sur un modèle client-serveur, qui nous donnera la possibilité ultérieure d'ajouter une interface web de gestion pour l'utilisateur final.

À cause de la complexité et de la richesse des infrastructures de gestion de clés, donc de leur implémentation au travers de newpki au sein de notre architecture, nous ne développerons pas ici la configuration complète du logiciel, mais uniquement son installation. Pour un accompagnement à la création et à la gestion de l'IGC, il suffit de se reporter à la documentation française officielle du logiciel [4].

Nous installons le client sur les postes des administrateurs. Pour ce faire, il suffit d'utiliser les dépôts Debian :

```
sudo aptitude install newpki-client
```

Nous faisons de même sur le serveur dédié à l'usage de l'IGC :

```
sudo aptitude install newpki-server
```

Lors de l'installation, un fichier `/var/log/newpki` est créé. Après l'initialisation, si tout s'est correctement déroulé, il contient les archives suivantes :

```
********************************************************************************
********************************************************************************

## {SOCKET server} > Sun Jul 1 22:08:16 2007# ADMIN : Starting socket server...
## {SOCKET server} > Sun Jul 1 22:08:16 2007# ADMIN : Socket server started
```

Le serveur écoute par défaut sur le port 3333. En connectant le client au serveur pour la première fois, il est nécessaire de se connecter avec le couple `root/root`.

L'interface nous permet de créer un à un les éléments qui composent notre IGC, comme nous pouvons le voir sur les figures 11.8 et 11.9.

Ainsi, nous allons d'abord créer l'entité PKI qui autorise l'administration de notre IGC. Viendront ensuite l'Autorité de Dépôt, chevron central, puis l'Autorité de Certification. À l'Autorité de Dépôt,

[4] http://sourceforge.net/docman/?group_id=50189

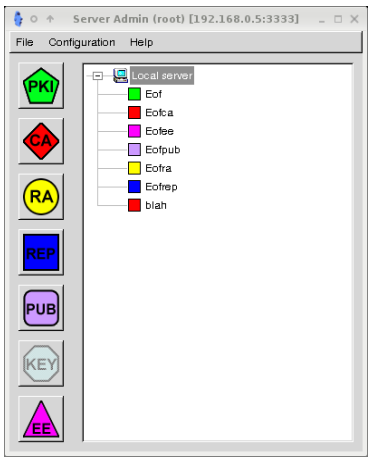


FIG. 11.8 – Création de l'IGC dans newpki

nous lions également l'Entité Finale, l'Autorité de Séquestre et l'Autorité d'Enregistrement. Notre IGC est complète et peut donc répondre aux demandes de certificats des utilisateurs finaux. Parmi les développements prévus, les certificats permettront aux personnes autorisées (enseignants, tuteurs, auditeurs) de se connecter à la plate-forme web de l'école. Cette authentification forte donnera accès aux contenus stockés, ainsi que l'accès au planning de formation en relation avec l'utilisateur.

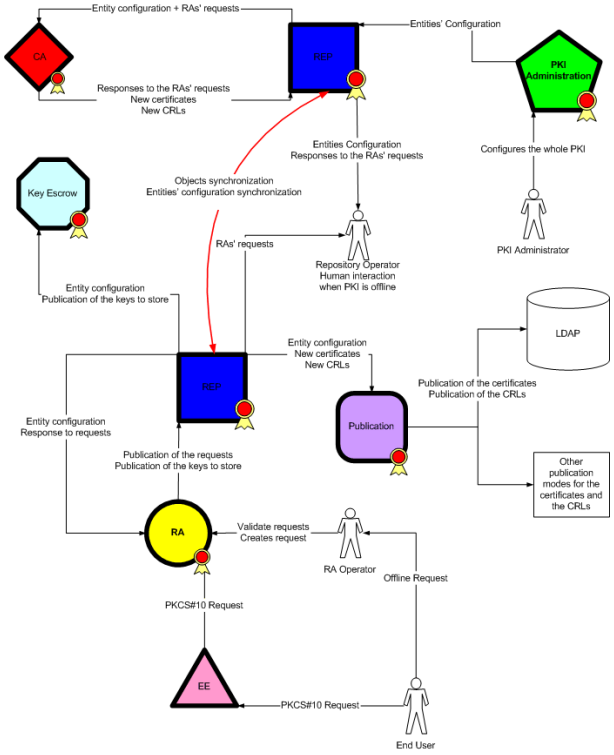


FIG. 11.9 – Principe de fonctionnement de newpki

Chapitre 12

Pot de miel

I Définition et usage

L'histoire se déroule il y a plus de 20 ans, pendant l'été 1986. Clifford Stoll vient de se voir confier par son supérieur la lourde tâche de trouver le logiciel qui a fait égarer 75 cents au Lawrence Berkeley Laboratory, Université de Californie. En réponse à un problème de conception, c'est en fait un utilisateur non autorisé qui apportera la réponse à Stoll. En effet, un pirate allemand, Markus Heiss, s'était introduit dans le système par l'intermédiaire d'une faille dans la fonction *sendmail* du logiciel *GNU Emacs*. La première action de Stoll fut de tenir un journal détaillé [32] des activités de son pirate, traces à l'appui, afin d'étudier son comportement et de comprendre quel était le ou les intérêts de ce dernier. Il s'avère que Hess s'était introduit dans *ARPAnet* à des fins d'espionnage militaire et cherchait tout fichier pouvant contenir les mots *nuclear*, *NORAD* ou encore *SDI*. Il en profitait également pour récupérer tous les mots de passe par l'intermédiaire d'un cheval de Troie. Afin de pouvoir localiser l'intrus, l'administrateur, à la recherche de ses trois quarts de dollar, crée un faux utilisateur, en la personne de *Barbara Sherwin*, soit disant secrétaire du *SDInet*, et met en place de nombreux leurres par l'intermédiaire de faux fichiers administratifs. Après de nombreuses collaborations avec les services spéciaux (NSA, FBI, CIA...) et opérateurs de télécom, Hess fut arrêté chez lui, en Allemagne occidentale. C'est la première fois qu'un pot de miel s'officialisera, ainsi qu'un certain nombre des techniques qui entourent cette technologie. Par la suite, c'est par l'intermédiaire du récit de Bill Cheswick, *An Evening with Berferd* [33], que continuera l'étude détaillée du comportement d'une équipe de pirates. Non seulement Cheswick étudia le comportement du pirate, mais mit également en place une interaction ainsi qu'un niveau de virtualisation dans son pot de miel. Bien que l'intervention de ces deux hommes et la force de leurs récits aient marqué la naissance de ce domaine, il y a eu de nombreux travaux dès lors, dont l'initiative remarquée de Lance Spitzner nommée *honeynet* [1]. Celle-ci regroupe les initiatives planétaires avec différents groupes d'études répartis dans les divers pays du globe.

Bien que nombreux et variés les honeypots peuvent être répartis selon deux grandes familles. Les premiers sont dits à *faible interaction*. L'idée est de logguer tout le trafic en offrant à l'attaquant un minimum de possibilités d'interagir avec l'intérieur, mais surtout l'extérieur. En effet, lors de la compromission d'une machine, de nombreux pirates cherchent à transformer leur nouvelle acquisition en machine de rebond pour attaquer de nouvelles machines, ce qui met l'administrateur du serveur en porte à faux vis-à-vis de la loi. Cependant, l'attaquant un peu expérimenté se rendra immédiatement compte de la supercherie et arrêtera de fait son attaque. Par conséquent, ce type de piège n'attirera donc que des

[1] http ://www.honeynet.org

script kiddies, c'est à dire des pirates débutants, ou des robots (virus, vers, etc). Un simple programme écoutant sur un port défini est donc en mesure de devenir un pot de miel. Le programme *netcat*, connu sous l'appellation *TCP/IP swiss army knife* [2] est en mesure de nous rendre ce service par la simple mise à l'écoute sur un port défini :

```
nc -l -p 80 > honeypot.log
```

Ce faisant, toute personne se connectant sur notre serveur pourra légitimement penser correspondre avec un serveur web. L'illusion ne durera pas, puisque lorsque une commande sera passée, aucune réponse ne sera donnée.

La deuxième famille est appelée *honeypots à forte interaction*. L'émulation de failles est remplacée par un véritable service vulnérable. Cela regroupe en général non seulement l'application, mais également tout le système d'exploitation qui la supporte. En effet, lors de la compromission d'une application, l'attaquant cherche en général à obtenir des droits supplémentaires pour travailler comme il lui plaît sur la machine et espérer cacher ses traces. De fait, disposer uniquement un logiciel faillible sur le système que l'on administre reviendrait, de manière imagée, à couper la branche sur laquelle nous nous trouvons. La machine entière est donc consacrée à la compromission, de l'utilisateur basique à l'administrateur. Si cela donne des résultats bien plus probants en matière de réponse à l'attaquant, les risques de perte de contrôle le sont également. De fait, il faut s'assurer que tout pirate ou *malware* attrapé dans notre pot de miel n'utilisera pas ce dernier pour attaquer d'autres cibles. Pour ce faire, de nombreuses techniques de firewalling liées à un IDS sont mises en jeu. Au niveau de la couche 3, on peut par exemple limiter le nombre de paquets sortant à une dizaine par heure, en autorisant a contrario toutes les tentatives de connexion. Pour être plus discret, on pourra placer un pare-feu en couche 2, ce qui aura pour effet de ne pas influer sur la durée de vie des paquets et d'autoriser une manipulation accrue, en rendant par exemple inoffensif tout paquet sortant, préalablement soumis à l'analyse de l'IDS.

S'il est facile de prendre connaissance des actions effectuées sur un honeypot à faible interaction, la chose s'avère plus difficile dans un environnement avec un vrai système. Pour cela, nous pouvons mettre en place des outils systèmes tels qu'un serveur syslog distant qui sera en mesure de recevoir toutes les informations d'événement système, un outil de détection d'intrusion, sur le réseau, qui pourra monitorer toutes les actions jugées à risque en partance ou en provenance de notre pot de miel. Le dernier outil qui peut s'avérer utile est le logiciel *sebek*, disponible sous forme de module noyau, qui récupère toutes les commandes tapées à l'insu du visiteur et les transmet via le réseau.

II Intérêt

On peut distinguer deux intérêts majeurs dans la mise en place de pots de miel au sein d'une infrastructure. Dans un mode de production, ceux-ci peuvent venir compléter les résultats des détecteurs d'intrusion, qui, malgré leur intérêt, rapportent encore aujourd'hui de nombreux faux-positifs et faux-négatifs. En effet, si chaque paquet est examiné par un IDS, le pot de miel, quant à lui, permettra une analyse rapide des logs et restituera une information condensée de ce qui se passe au sein de l'infrastructure. En cas de nouveau virus ou nouvel exploit, l'analyse de ces logs permettra d'incorporer les signatures dans l'IDS. Dans ce cas d'utilisation, il faut placer le pot de miel à côté des serveurs de production. L'effet attendu est également d'attirer le pirate potentiel vers le honeypot plutôt que vers les serveurs sensibles.

[2] Tiré du manuel du logiciel.

Le deuxième intérêt que l'on trouve dans la mise en place de pots de miel est un aspect de recherche. Bien que, comme nous le montre l'histoire, les premiers projets sont apparus dans un contexte de production, un consensus s'est rapidement créé autour de l'étude comportementale et technique que l'on pourrait obtenir à partir de ces appâts. En outre, on espérait comprendre les buts des pirates, mais aussi découvrir de nouvelles failles de sécurité, de nouvelles manières de les exploiter. De tout cela, on peut déduire les intentions du pirate : uniquement l'obtention de ressources ou une cible bien précise, ou encore l'exploitation massive d'un logiciel grandement déployé pour créer un *botnet*.

III Mission : SSH

1. Définition de la cible

Le pot de miel que nous avons installé au sein de l'architecture trouve un intérêt médian par rapport à ce que nous venons de présenter. En effet, il sert à la fois à détourner et engluer les hypothétiques pirates dans un piège, mais nous permet également d'étudier leur comportement, détecter de nouveaux robots et, pourquoi pas, découvrir un nouvel exploit. Pour cela, nous avons choisi d'installer sur un serveur dédié, un honeypot à faible interaction . Ce dernier émule un serveur SSH avec un grand nombre de comptes possédant plusieurs mots de passe faibles. Par exemple, on pourra tout aussi bien accéder au compte administrateur grâce au mot de passe `root` que `password`. Ceci n'est qu'un exemple puisqu'il n'y a pas moins de 7294 mots de passe associés à l'utilisateur *root*. Le choix du niveau d'interaction permet de ne pas avoir une vigilance de chaque minute et limite aussi grandement le risque de voir un éventuel pirate détourner notre machine pour pénétrer plus avant notre infrastructure ou aller visiter d'autres serveurs.

2. Mise en place du pot de miel

Pour mettre en place notre pot de miel, nous avons configuré un serveur GNU/Debian stable puis ajouté le logiciel *Kojoney* [3]. Celui-ci est un petit programme écrit en python, qui permet de simuler un environnement de type terminal par l'intermédiaire d'un serveur SSH. Les attaques sur le service SSH sont assez nombreuses, car en cas de succès de la tentative d'intrusion, un shell est immédiatement disponible, donnant accès à de nombreuses informations et possibilités d'action.

a. Installation de Kojoney

Kojoney est un logiciel écrit en python, il nécessite donc l'interpréteur de commande du même nom. Après avoir téléchargé le *tarball*, il suffit de lancer le script d'installation pour que le logiciel soit fonctionnel. Un certain nombre de modules complémentaires sont également installés, soit pour permettre le fonctionnement du service, soit pour compléter le rapport d'analyse dont nous reparlerons un peu plus bas. L'étape finale achève le processus en demandant si le pot de miel doit être démarré au lancement du serveur.

```
# sh INSTALL.sh
Kojoney Honeypot installer.

Press enter to view the license agreement ...
```

[3] http ://kojoney.sourceforge.net/

```
<<< NOTE: After read the license agreement press 'q' to exit >>>
Do you accept the ZPL, MIT and GPL license terms (yes/no) ?
yes
All licenses accepted.
****************************************
 Kojoney Honeypot Installer version 0.0.3
****************************************

Step 1 - Copying files
Step 2 - Building libraries
 [+] Building and installing [IP-Country]
 [+] Building and installing [Geograpy-Countries]
 [+] Building and installing [Zope Interfaces]
 [+] Building and installing [Twisted extension]
 [+] Building and installing [PyCrypto]
 [+] Building and installing [Twisted Conch extension]
Step 3 - Installing documentation
 [+] Installing man pages
Step 4 - Changing permissions and creating symbolic links
 [+] Creating symlinks
Step 5 - Final questions and fun
Kojoney installation finished.
```

Avant de pouvoir lancer notre logiciel, il est intéressant de faire un tour dans sa configuration, pour pouvoir configurer le port sur lequel écoutera l'application, ainsi que l'endroit où seront stockés les logs :

```
ROOT_CONFIG_PORTS = [110]
ROOT_CONFIG_LOGS = [sys.stderr, open("/var/log/honeypot.log", "a")]
DOWNLOAD_REAL_FILE = True
DOWNLOAD_REAL_DIR  = "/var/log/kojoney/"
```

Le port 110 est sélectionné afin de ne pas interférer avec le véritable service ssh qui tourne sur la plate-forme. Ce port sera de toute façon redirigé sur le port 22, port par défaut du service SSH, au travers d'un *NAT* sur le firewall d'entrée de site. De ce fait, l'attaquant sera persuadé de joindre directement un véritable serveur SSH.

Le `ROOT_CONFIG_LOGS` pointe le fichier dans lequel seront ajoutés tous les événements de connexion sur le pot de miel. Les `DOWNLOAD_REAL_FILE` et `DOWNLOAD_REAL_DIR` permettent quant à eux de définir où seront stockés les fichiers que tentent de télécharger les attaquants, et ce afin d'étude ultérieure.

3. Étude de l'impact

Durant les quelques temps où le pot de miel a fonctionné, nous avons eu quelques visites, ce qui permet une petite analyse de l'activité et du comportement des pirates. Les différents exemples suivants partent du rapport effectué via kojoney.

Lors du relevé des visites (tableau 12.1), les identifiants les plus utilisés se sont révélés être `root` et `admin`. Ceci s'explique du fait que ces comptes ont des privilèges plus élevés et sont donc prisés par les

intrus. Quant au reste des utilisateurs ayant un nombre de connexions similaire, on peut extrapoler que ces connexions sont réalisées par des robots possédant des dictionnaires de mots de passe.

TAB. 12.1 – Liste des utilisateurs authentifiés avec succès

Nombre de connexions	Compte
20	root
11	admin
5	richard
4	web
4	user
4	test
4	sales
4	pgsql
4	mail
4	guest
3	wwwrun

Comme nous avons pu le voir lors du chapitre sur SSH, il est possible de se connecter via une authentification par clé publique. Nous avons pu identifier 24 tentatives qui ont échoué.

Lorsque des tentatives ont abouti, par l'intermédiaire de clé ou de mot de passe, les commandes remontées par les logs montrent que le listing de l'arborescence du serveur est en tête, l'interrogation sur les connectés, le changement de répertoire, la récupération de logiciels, suivi par la connaissance de l'environnement matériel (cpu et carte réseau).

TAB. 12.2 – Commandes exécutées par les intrus

Usage	Commande
16	ls
7	w
6	cd var
5	ls -a
5	cd tmp
4	wget
4	cat /proc/cpuinfo
4	/sbin/ifconfig -a | grep inet
3	uname -a
3	pwd
3	cd scan
2	quit
2	ls a-

Le tableau 12.3 relate les tentatives de connexion. Sur un nombre de 787 relevées, 43.9% viennent de Chine et 49.3% proviennent de Corée. L'Asie est donc bien majoritaire dans les tentatives de piratage auxquelles nous avons eu à faire. On constate également que les pirates se dispersent.

Cependant, l'analyse suivante nous montre que seuls des européens (français et roumains) ont

TAB. 12.3 – Attaquants et pays d'origine

Adresse de l'attaquant	Nombre de tentative(s) de connexion(s)	NIC	Pays
221.0.130.43	162	CN	Chine
211.43.220.66	384	KR	Republic of Korea
210.209.130.230	3	TW	Taiwan Province of China
200.60.74.14	1	PE	Peru
200.161.31.82	1	BR	Brazil
193.251.47.229	1	FR	France
192.168.0.1	3	**	Intranet address
172.178.213.150	2	US	United States
90.28.24.244	1	FR	France
86.64.63.100	9	FR	France
86.107.28.191	2	RO	Romania
84.112.63.16	5	AT	Austria
80.53.103.50	28	PL	Poland
61.96.94.29	4	KR	Republic of Korea
61.56.132.11	4	TW	Taiwan Province of China
61.51.17.161	9	CN	China
61.31.239.107	168	TW	Taiwan Province of China

mené l'attaque jusqu'au bout et ont fait suite à l'attaque, en utilisant le shell mis à disposition :

```
Humans detecteds by IP
----------------------
Typo error filter: Human detected at 192.168.0.1 (**, Intranet address)
Typo error filter: Human detected at 90.28.24.244 (FR, France)
Typo error filter: Human detected at 86.107.28.191 (RO, Romania)
Typo error filter: Human detected at 86.64.63.100 (FR, France)

4 human(s) total
```

Chapitre 13

IPv6

I Pourquoi utiliser IPv6

Actuellement, le protocole de traitement de la couche réseau est *IPv4*, pour *Internet Protocol version 4*. Ce protocole a été développé par Vinton Cerf suite à une demande du DOD, pour ARPAnet. Plus de trente ans ont passé et un certain nombre de limitations sont apparues. Tout d'abord, le nombre d'adresses IP disponibles est limité à 4 294 967 296, soit 2^{32}. Cette restriction, lors de la création du protocole, semblait ne pouvoir jamais être atteinte. Cependant, l'évolution de l'informatique, l'explosion de l'usage Internet et la naissance du tout embarqué créent une sérieuse remise en question de ce pool qui pouvait être considéré comme acquis. Ainsi, en 1992, après l'ouverture commerciale d'Internet, il fallut, une année plus tard, déclencher un plan d'urgence, puisqu'il ne restait plus aucune classe B disponible. Ces mesures se concrétisèrent en deux points :

- la création de la notation CIDR et sa mise en place [34]. Le résultat fut une diminution du gaspillage de l'espace d'adressage, ainsi que la possibilité d'agrégation, ce qui entraîne une réduction de la taille des tables de routage.
- la création et la mise en place d'un plan d'adressage privé [35] et du NAT [36] (traduction d'adresse réseau).

Ces mesures palliatives ne sont que temporaires. De fait, les mesures techniques induisent des contraintes et de nouveaux problèmes. Les protocoles dynamiques, tel que le FTP, doivent être traités indépendamment. Une couche de sécurité supplémentaire est obligatoire pour assurer l'intégrité et la confidentialité. L'humanité continue de croître, les applications technologiques aussi. Il est donc essentiel, dès ce moment là, de travailler sur le successeur d'IPv4, en tâchant de résoudre les manques inhérents à ce dernier. C'est en 1995 que l'IETF [1] fournira une première mouture d'IPv6, avant de finaliser une seconde version fin 1998 [37].

[1] http://www.ietf.org

II Fonctionnalités d'IPv6

1. Principales caractéristiques d'IPv6

a. Espace d'adressage

S'il n'y a qu'une seule chose à retenir, c'est qu'un nouveau plan d'adressage est défini. Il y a maintenant 128 bits disponibles, soit une réserve de 2^{128} adresses, soit $3,40282366921x10^{38}$. Chaque adresse est divisée en 8 blocs de 16 bits séparés par le caractère « : ». Contrairement à IPv4, qui utilisait une notation décimale, c'est le format hexadécimal qui a été retenu, c'est à dire une valeur composée entre *0* et *ffff* inclus. S'il existe des blocs nuls, c'est à dire notés à *0*, on peut les omettre, et ce de manière consécutive. La nouvelle notation se voit alors composée de « :: ». Ceci n'est cependant possible qu'une fois par adresse, car c'est une source d'ambiguïté.

Voici un exemple d'adresse IP réelle :

```
host -t AAAA  www.jp.freebsd.org
2001:2f0:104:1:2e0:18ff:fea8:16f5
```

Si nous avions eu l'adresse `2001:0000:0000:0000:0000:0000:0000:16f5`, il nous aurait été possible d'écrire l'adresse sous la forme `2001:0:0:0:0:0:0:16f5`, ou encore `2001::16f5`. Toutefois, si l'adresse avait été de la forme `2001:0:0:1:0:0:0:16f5`, nous aurions seulement pu écrire `2001::1:0:0:0:16f5` ou `2001:0:0:1::16f5` et pas `2001::1::16f5` afin de lever l'ambiguïté.

b. Type d'adresse

TAB. 13.1 – Découpage d'une adresse IPv6

0 15	16 31	32 47	48 63	64 127
Espace d'allocation	Fournisseur d'accès	Client	Réseau	Identifiant

Rémi Denis-Courmont, auteur d'une implémentation libre des tunnels Teredo, Miredo [2], propose l'analyse des adresses IPv6. Pour notre étude, nous proposons l'adresse IPv6 en notre possession :

```
Décomposition d'une adresse IP

Adresse IPv6 2001:0:53aa:64c:0:fbff:ad0f:301f
Type (RFC 3513) Unicast global
Espace/Allocation (RFC 3513) Unicast global (2000::/3)
Catégorie Internet définitif
Encapsulation Teredo tunneling
Préfixe Teredo 2001:0000
IPv4 du serveur Teredo 83.170.6.76 (miredo.svr02.mucip.net)
IPv4 externe du client Teredo 82.240.207.224:1024
(home.gaulier.info:1024)
Flags Teredo 0....... ........ = Restricted
Sous-réseau (bits 48 à 63) 064c
```

[2] http://www.remlab.net/miredo/

```
Identifiant d'interface (bits 64 à 127) 0000:fbff:ad0f:301f
Type d'identifiant Statique ou aléatoire
```

L'analyse de l'adresse montre que nous sommes en possession d'une adresse directement accessible sur Internet, mais également que nous utilisons un tunnel Teredo. En effet, peu de fournisseurs d'accès français mettent à disposition une connectivité native IPv6. Le protocole Teredo nous permet d'encapsuler des paquets IPv6 dans des datagrammes UDP, sur une couche IPv4, derrière un NAT.

TAB. 13.2 – Type d'adresse

Type d'adresse	Préfixe binaire	notation IPv6
Non spécifié	00...0 (128 bits)	: :/128
Loopback	00...1 (128 bits)	: :1/128
Multicast	11111111	FF00 : :/8
Lien local unicast	1111111010	FE80 : :/10
Site local unicast	1111111011	FEC0 : :/10
Unicast global	(everything else)	b

c. Auto-configuration

Les adresses sont à présent complexes. Il est difficile de les retenir, il est donc difficile de les allouer, sans évoquer le temps nécessaire pour mettre en place chaque configuration. Dans cette optique, les matériels doivent être *plug & play*, c'est à dire supporter l'auto-configuration, afin de permettre l'attribution automatique d'une adresse. Pour ce faire, lors de l'activation de la carte réseau, le client demande les paramètres réseaux : préfixes, routeur par défaut, limite du nombre de sauts (hop limit)… Les équipements peuvent obtenir automatiquement une adresse IPv6, seuls les routeurs doivent être configurés à la main (pour l'instant) mais les adresses ne sont pas enregistrées automatiquement dans le domaine DNS, il est nécessaire de faire appel à un protocole externe, le *DNS Dynamic Update*.

d. Multi-homing et mobilité

Une fonction importante et intéressante prise en compte dès la conception du protocole consiste dans la conservation d'une même adresse IP quel que soit le réseau auquel est connecté l'équipement. Avec IPv4, quelques notions de configurations logicielles étaient présentes, avec IPv6, tout est dans la pile. Pour réaliser cela, le nomade reçoit une adresse déterminée comme constante, appelée *mère*, issue de son réseau initial. À chaque déplacement, une adresse temporaire, nommée *adresse temporaire*, issue du réseau visité est également attribuée à l'interface. D'un point de vue IP, c'est l'adresse temporaire qui permet la connexion, sauf lorsque l'on est connecté au réseau initial, alors que pour les protocoles de plus haut niveau, c'est l'adresse mère qui leur sert d'interface.

Le collectif Gisèle Cizault [38] relate les trois points cruciaux liés à la mobilité :
- pouvoir communiquer ;
- être joignable ;
- conserver les communications en cours lors des déplacements.

Le mécanisme de multi-homing permet de répondre grandement à cette problématique.

III Sécurité

Le travail principal autour de la sécurité s'est concentré sur deux axes : l'authentification et le chiffrement. Le déploiement du protocole IPv4 étant assez répandu, il était important de pouvoir utiliser les mécanismes de sécurité à la fois pour les versions 4 et 6 d'IP. Le résultat de ce travail est le protocole *IPsec*. Face aux problèmes de légalité vis-à-vis de la cryptographie, l'authentification et le chiffrement devaient être séparés de bout en bout ou par tunnel.

Ce protocole, longuement réfléchi se base sur différents mécanismes. Ainsi, on retrouve du chiffrement symétrique et asymétrique, un nouveau champ d'entête, *AH* et un nouveau champ de contenu, *ESP*, un changement de taille, également, puisque *IP* passe du mode non connecté à connecté. Nous reviendrons sur ce point précis un peu plus loin. Il est important de noter qu'on autorise l'utilisation d'un algorithme de chiffrement vide. Ce dernier permet à ceux qui ne désirent pas dépenser du temps de calcul d'employer tout de même *IPSEC*.

1. AH

L'*Authentication Header* prend sa place au sein de l'entête *IP*. Il décale l'entête du protocole encapsulé dans *IP*, par exemple *TCP* (6) ou *UDP* (17), pour insérer le sien (51), ainsi que les champs qui l'accompagnent. Le dessin suivant vous en dira davantage :

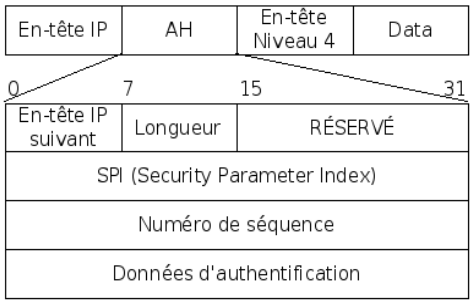


FIG. 13.1 – En-tête AH

Examinons de plus près les champs de *AH* :

- le SPI contient entre autre un pointeur qui indique à l'hôte destinataire quelle clé partagée doit être utilisée pour l'échange,
- le numéro de séquence est unique, même si un paquet est retransmis. Cela permet de limiter les attaques dîtes de rejeu,
- les données d'authentification sont le champ qui peut être considéré avec le plus d'attention. Pour consituer l'*ICV* (Integrity Check Value), il se base sur les champs immuables du paquet (pas le TTL par exemple), sur les champs prédictibles (comme l'adresse destination), sur certains champs de l'entête AH, sur les données du paquet IP. On calcule le tout sous forme d'un *HMAC* (Hashed Message Authentification Code), ce qui permet d'obtenir une empreinte sûre du message.

Ce procédé permet donc de s'assurer de l'intégrité d'un paquet transmis ainsi que de sa provenance. Il ne permet cependant pas la confidentialité des données qu'il contient. De plus, *AH* est antérieur à ESP. Son mode de calcul oblige à bufferiser le paquet pour calculer l'*ICV* et le réinjecter dans l'entête, ce qui défavorise le taux de transfert des paquets.

Pour terminer, *ESP* recoupe au fil du temps un certain nombre de fonctionnalités de *AH*. À l'origine, *AH* gérait l'authentification et *ESP* le chiffrement. Ce n'est plus le cas actuellement. Il y a fort à parier que, dans les années qui viennent, *AH* disparaîtra de fait. En attendant, ses défenseurs soulignent que *AH* fait partie de l'entête *IP*, ce qui est, disons plus standard et qu'il ne comporte pas de chiffrement, ce qui lui permet une exportation mondiale plus aisée.

2. ESP

ESP pour *Encapsulated Security Payload* a pour fonction, tout comme *AH*, de garantir l'intégrité du paquet, assumer l'authentification de la provenance et prévoir un mécanisme contre le rejeu. Jusque là, rien de nouveau. Ce qui apporte un grand plus à cette méthode, c'est le chiffrement du contenu du paquet, tout comme vous pouvez le voir dans la figure 13.2.

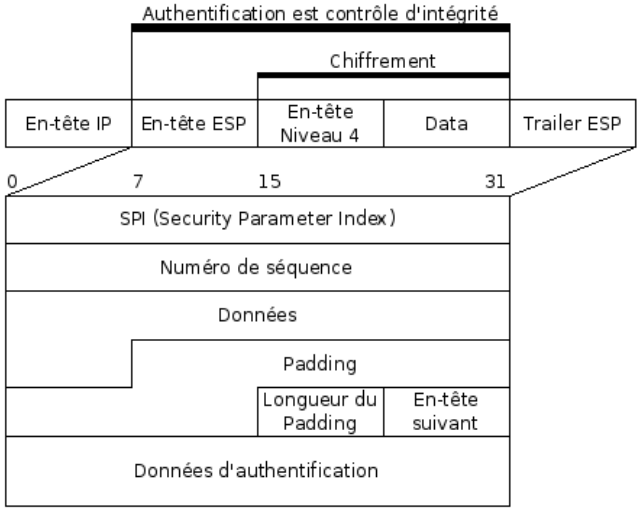


FIG. 13.2 – ESP

Les données d'authentification sont calculées sur les champs d'*ESP* via l'*ICV* expliqué précédemment. Le champ *Padding* permet de «bourrer» afin de satisfaire aux exigences longueur pour les algorithmes de chiffrement.

3. Les particularités du SPI

Nous l'avons vu dans les deux modes précédents, le champ *SPI* est une composante essentielle. Revenons donc quelques instants sur son contenu un peu plus précisément. Tout d'abord, l'objectif de ce SPI est d'identifier de manière unique une *SA* (Security Association). Elle définit une connexion de type simplex, c'est à dire dans un sens unique, d'un émetteur vers un récepteur. Pour organiser un dialogue, il faudra donc nécessairement deux *SA*.

La *SA* se compose des éléments suivants :
- un compteur qui permet de générer les numéros de séquence,
- un drapeau qui indique si il y a dépassement du compteur,
- une fenêtre d'anti-répétition qui doit contenir le prochain numéro de séquence,
- les informations sur AH : algorithme d'authentification, clé, durée de vie, ...,

– les informations sur ESP : algorithme d'authentification et de chiffrement, durée de vie, vecteur d'initialisation, clé, ...,
– un indicateur sur le mode *IPSEC* utilisé,
– la durée de vie de la *SA*,
– le MTU.

Nous parlons du drapeau qui indique s'il y a un dépassement du compteur. Nous avons précisé, plus haut, que chaque numéro de séquence doit être unique. Ainsi, si les 2^{32} possibilités sont épuisées, la connexion est interrompue pour créer une nouvelle *SA*.

4. Mode transport et mode tunnel

Dans la définition de la *SA*, nous avons évoqué un indicateur du mode *IPSEC* employé. La figure 13.3 présente le mode transport (en haut) et le mode tunnel (en bas). Nous n'avons représenté ici que le mécanisme *ESP*.

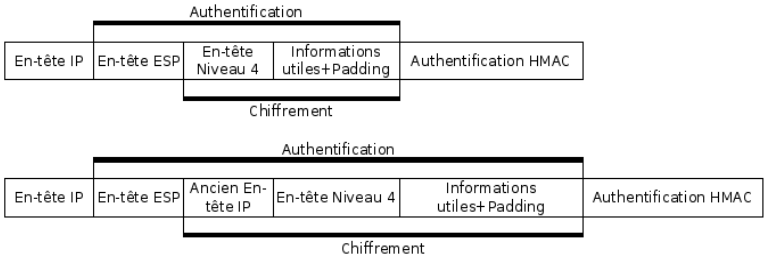


FIG. 13.3 – ESP en mode transport et ESP en mode tunnel

Le mode transport est celui présenté ici par défaut, puisqu'il s'agit d'insérer les mécanismes *AH* ou *ESP* entre l'entête IP et l'entête du protocole de niveau 4, comme nous l'avons déjà vu. Dans le mode tunnel, la totalité du paquet IP est encapsulé dans un nouveau paquet IP. Ceci peut s'avérer très utile, lorsque vous devez par exemple traverser des pare-feux, puisque la destination finale est placée derrière ce dernier. Si vous n'utilisez pas cette méthode, vous ne pourrez pas traverser le pare-feu et établir une connexion avec votre destination initiale.

5. La gestion des clés

Dès lors que l'on parle de chiffrement, on sous-entend la notion de clés, que celles-ci soient publiques, privées ou partagées. Ainsi, la gestion des clés est remise au protocole ISAKMP, alors que l'échange de clés se base sur IKE.

a. ISAKMP

Comme dans tout établissement de connexion chiffrée, il faut établir quels seront les algorithmes de chiffrement utilisés. On peut bien sûr le faire dans un cadre restreint, comme pour une petite infrastructure, mais si l'on passe à un peu plus grand, par exemple Internet, cela devient beaucoup plus complexe. C'est pour cela que l'on a mis au point ISAKMP, qui va tout d'abord définir les moyens de sécurisation des échanges qui vont suivre, puis mettre en place les caractéristiques des futures connexions (SA). On peut considérer que cette définition prend un temps important, mais lors de nouvelles connexions ou reconnexions, tout aura déjà été paramétré et ce temps deviendra alors un gain.

b. IKE

Internet Key Exchange, basé sur Diffie-Hellmann, authentifie les différents protagonistes de l'échange en vérifiant de manière croisée les clés publiques de chacun (clé publique de A, clé privée de B et inversement), puis se charge de la génération d'une clé partagée, utilisée pour le chiffrement symétrique.

Il est à retenir que dans sa première version, *IKE* est un protocole peu sûr. Une deuxième version d'*IKE* a vu le jour et améliore sur un certain nombre de points ce dernier (gestion du NAT, réduction des échanges initiaux, réduction des états d'erreurs, meilleure robustesse du protocole, ...).

IV Comment se relier au réseau IPv6 ?

Il n'existe à ce jour aucun fournisseur français pour l'ADSL grand public qui offre une connectivité IPv6. Pour les entreprises, seul le fournisseur Nerim propose actuellement ce service dans son catalogue ; Orange teste une expérimentation sur un parc restreint. Autant le dire, pouvoir se connecter en France n'est pas une affaire aisée, à moins d'avoir une entrée dans le réseau *RENATER* [3]. De ce fait, nous devions passer par une solution de contournement :

- un tunnel statique,
- un tunnel 6to4,
- un tunnel de type *Teredo* ;

Le tunnel statique nécessite de posséder deux extrémités. Une de notre côté qui fait circuler le trafic, une autre pour nous relier à un réseau complètement maillé en IPv6. Malheureusement, aucune entrée n'est dédiée au grand public, nous n'avons donc pas pu réaliser cette expérience.

La mise en place d'un tunnel dit « 6to4 » est une bonne solution. Elle profite de l'adresse IPv4 allouée par le fournisseur d'accès pour transporter des paquets IPv6 encapsulés. Pour cela, une adresse IPv6 est forgée avec comme préfixe `2002`, suivi de l'adresse sous forme hexadécimale du routeur de transport ainsi que de l'adresse mac de la machine. Ce procédé est tout à fait utilisable, cependant aucun mécanisme de filtrage n'est possible sur les méthodes proposées à ce jour. De plus, la question du NAT n'est pas gérée.

La dernière solution qui s'offre à nous est la mise en place d'un tunnel de type *Teredo*, défini dans la RFC 4380 [39]. Le transport des paquets de version 6 s'effectue sur le protocole UDP de la version 4, ce qui permet d'échapper au problème du NAT. Rémi Denis-Courmont [4] est l'auteur d'une implémentation libre de ce protocole nommée *miredo*. Pour installer le logiciel, rien de plus simple :

```
aptitude install miredo
```

Le logiciel crée une interface dédiée durant ce process :

```
teredo    Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-
  00-00-00-00-00-00-00-00
          inet6 addr: fe80::ffff:ffff:ffff/64 Scope:Link
          inet6 addr: 2001:0:53aa:64c:0:fbff:ad0f:301f/32 Scope:Global
          UP POINTOPOINT RUNNING NOARP  MTU:1280  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
```

[3] Réseau National de télécommunications pour la Technologie l'Enseignement et la Recherche : http://www.renater.fr
[4] http ://www.remlab.net

```
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 b)  TX bytes:144 (144.0 b)
```

Il ne reste plus qu'à tester que la connectivité IPv6 est bien au rendez-vous :

```
jop@ema:~ ping6 -c 4 www.kame.net
PING www.kame.net(orange.kame.net) 56 data bytes
64 bytes from orange.kame.net: icmp_seq=1 ttl=52 time=1363 ms
64 bytes from orange.kame.net: icmp_seq=3 ttl=52 time=483 ms
64 bytes from orange.kame.net: icmp_seq=4 ttl=52 time=484 ms

--- www.kame.net ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3285ms
rtt min/avg/max/mdev = 483.995/777.420/1363.624/414.508 ms, pipe 2
```

La connexion est établie. Notre requête s'effectue en IPv6, puis est encapsulée dans des datagrammes UDP. Pour sortir de notre réseau, nous empruntons le réseau IPv4, à destination du routeur Miredo configuré par défaut dans le logiciel. De ce routeur, nous repassons en IPv6 pour atteindre le site kame.net.

La connectivité de notre plate-forme est donc prête à assumer un passage en IPv6, cependant la connectivité française proposée par les fournisseurs d'accès et le nombre de réseaux pairs capables de partager cette connectivité sont encore restreints. Il est important de garder constante la veille sur l'évolution des réseaux IPv6, mais il est sûrement encore trop tôt pour proposer cette solution en production. De plus, de récentes failles de conception et d'implémentation [40] amènent à désactiver les fonctions IPv6 dès l'entrée de la plate-forme.

Chapitre 14

Analyse et bilan

Avant de conclure sur cette étude, je souhaitais revenir sur quelques points de la plate-forme et de ce travail. Celui-ci se veut conséquent, mais il n'est en aucun cas exhaustif et reste perfectible. Son perfectionnement passe avant tout par l'approfondissement de chacune des solutions techniques proposées, des scripts de suivis et de sauvegarde, de la supervision. On ne peut malheureusement pas tout traiter en si peu de pages sans négliger les aspects primordiaux, qui sont ceux sur lesquels je pense m'être focalisé.

Voici quelques retours sur des points que je n'ai pu aborder, mais qu'il semble intéressant de proposer en tant que pistes pour de futurs approfondissements.

Notre architecture réseau ne traite que peu de redondance sur défaillance, ou du moins, de plan de recouvrement. En effet, que se passe-t-il si l'on perd le pare-feu, le serveur virtuel ou encore le répartiteur de charge ? Plusieurs possibilités sont offertes :

- la mise en place de matériel en redondance logicielle (*VRRP*, *Heartbit*, *IPMP*…),
- la disponibilité de *spare* qui permet une restauration dans un délai rapide.

Il m'a paru important de traiter les schémas initiaux et essentiels que peu d'ouvrages de l'art abordent, comme par exemple le mécanisme de création et de gestion d'un pare-feu, au détriment, peut-être, de nouveaux éléments techniques qui sont communs et par conséquent plus documentés dans le cadre de déploiement de plate-forme. Quant à l'option matérielle, notre condition de *laboratoire de tests et de recherche* n'imposait pas comme contrainte première un schéma de disponibilité critique. Lors de la mise en production, il est donc primordial de prévoir du matériel supplémentaire à l'achat, ainsi que d'inclure la gestion de la haute-disponibilité sur tout élément critique de l'architecture.

Nous n'avons pas, non plus, présenté d'architecture *n-tiers*. En effet, les applications (sites institutionnels statiques) ne s'y prêtent pas. Cependant, dès lors qu'une base de données ou un frontal au serveur web devient une obligation, il faut rajouter des zones hermétiques, filtrées par le pare-feu, et n'autoriser qu'un trafic bien appréhendé. Il en découle un corollaire immédiat : les administrateurs doivent travailler au plus proche des développeurs, et ces derniers doivent fournir au plus tôt les dossiers d'architecture technique qui définissent les contraintes techniques d'espace, de système et de réseau. Ces discussions sont encore difficiles à établir, il faut donc aborder les choses sous l'angle de procédures qui permettent à chacun de savoir de manière organisée ce qu'il a à faire.

Pour en revenir à des éléments plus techniques, nous aurions pu poursuivre la sécurisation du DNS en comparant DNSsec avec IPsec, ou encore nous intéresser à de nouveaux mécanismes de transferts de zones via la copie sécurisée (scp) ou la synchronisation distante (rsync). Pour le serveur web, il y a

également un enjeu sur la gestion des masques et des expressions régulières en relation avec les URLs, travail effectué par les reverse-proxies. Cependant, ce sont des sujets qui font les gorges chaudes pendant les conférences de sécurité, mais qui n'apportent pas de réelles avancées dans ce domaine. On reste sur des réponses palliatives, sans pouvoir proposer du curatif. Il est important de continuer la veille sur ces sujets, car les nombreux travaux présentés chaque année permettent de s'approcher d'un traitement plus satisfaisant.

Sur cette même idée, l'approche conjointe de la virtualisation et des pots de miel ne peut qu'avancer, ce qui permettra d'accroître le nombre de solutions et par là même de varier les approches d'utilisation.

Il semble important de souligner que les solutions logicielles proposées dans ce mémoire correspondent aux critères suivants :

- standards du marché (1),
- logiciel le plus avancé parmi ses pairs (2),
- design et administration en conformité avec la plate-forme (3),
- parfaite adaptation aux besoins de la démonstration (4).

Bind, Apache, Syslog-ng, Netfilter et OpenSsh concordent avec (1). Ils se sont imposés par leur qualité, leur gratuité, à la réactivité et à la disponibilité des équipes de développeurs des produits respectifs. Leur licence libre est un atout supplémentaire non négligeable. Le gestionnaire de clés rentre dans (2). C'est en effet un domaine encore vierge, où beaucoup de choses restent à défricher, mais où l'usage est encore restreint aux initiés. Les serveurs virtuels (3) font face à de nombreuses discussions, d'autant plus avec le rachat de XenSource par la société Citrix. Pour notre part, la solution *Vserver* correspond véritablement aux besoins exprimés par l'école. Pour finir, dans le cadre de ce mémoire, les logiciels de pot de miel et de gestion de tunnels IPv6 étaient en adéquation avec (4).

Des sujets déjà présents au sein de l'école n'ont pas du tout été traités : c'est le cas de la plate-forme de courrier, qui gère les mécanismes d'anti-spam, de listes blanches, grises et noires, d'antivirus. Cette plate-forme fonctionne depuis trois années et il faudrait y consacrer un mémoire complet afin de pouvoir avoir un regard intéressant sur la question.

Pour terminer ce bilan, je souhaite revenir sur ce qui me parait le plus important : le métier de l'administrateur est d'abord de garantir que les services nécessaires pour les utilisateurs soient disponibles. C'est à cet égard qu'il doit veiller à leur sécurité :

- externe, cela va de soi,
- interne.

Par exemple, puisqu'on sait que les utilisateurs ne font pas toujours attention ou ne s'en préoccupent pas, il est possible *forcer* des modes d'utilisation comme une redirection automatique de *http* vers *https* dès lors qu'il y a une page d'authentification. Pour l'utilisateur, l'usage est transparent et cela garantit certaines transactions. L'éducation est la force première, mais l'accompagnement est long. Il faut donc s'assurer qu'en cas de manquements, il y aura toujours un filet de protection.

Conclusion

Voilà le moment tant attendu, il est temps de conclure, non en signifiant la fin d'un projet, mais simplement en marquant le temps de rupture adéquat d'une étape vers une concrétisation.

L'étude de la mise en place d'une solution de ressources réparties et sécurisées au sein d'une infostructure a été un sujet passionnant, tant sur le plan humain, de par les échanges avec mon tuteur, Alix Mascret, que sur le plan technique. Enrichissant, jour après jour, mon bagage, ma connaissance du système d'exploitation en général, linux en particulier, mais également de réseau et de sécurité. Ainsi, l'environnement des logiciels libres que je pouvais déjà côtoyer, s'avère riche et exceptionnellement prolifique, tant par les « monsieur tout le monde », contributeurs des différents projets que par son exubérance technologique et technique.

Les produits logiciels découverts et présentés tout au long de ce mémoire peuvent être considérés comme technologiquement mûrs. En effet, de nombreuses entreprises s'en font déjà écho, depuis de nombreuses années. C'est entre autre le cas d'Apache, Bind ou OpenSSH. Ces solutions n'en sont plus au stade de la recherche ou de la tentative, mais s'imposent en seigneurs du genre. On pourra se référer aux différentes statistiques issues du marché pour s'en assurer. C'est sûrement grâce à ces expériences réussies que le positionnement dans des secteurs de mise en production, pour les gestionnaires de clés centralisées ou de recherche, pour les pots de miel, peut voir le jour avec suffisamment d'acteurs crédibles pour les soutenir et les employer. Si les premiers ont leur avenir assuré en entreprise, pour des raisons de sécurité, de centralisation et de facilité de gestion, que peut-on dire des seconds ? Ils restent pour le moment l'apanage de chercheurs spécialisés car les limites sécuritaires peuvent être vite atteintes. Il faut, au préalable, dessiner clairement les stratégies que l'on souhaite pour ce type d'outil, fixer les buts et définir les moyens mis à disposition. Sans ces *garde-fous*, il y a un risque non négligeable de compromettre son architecture et sa crédibilité.

Pour ce qui fut de l'intégration d'IPv6, je me dois de prendre un temps, à part, pour parler de son intégration au projet. Ce protocole a déjà dix années derrière lui. On en vient, progressivement à son intégration, même si de nombreux leaders du marchés, à commencer par les équipementiers réseaux sont encore en retrait face à cette évolution. Si l'on a pu voir les avancées technologiques de ce protocoles face à son prédécesseur, ainsi que l'intégration qu'il a pu être possible de mettre en place, on ne peut négliger l'absence d'utilité à ce jour de celle-ci. C'est bien pour cela qu'il faut souligner le caractère prévisionnel de cette partie de travail. Pas inutile, loin s'en faut, du moins dans le temps. *Ready for IPv6*, comme savent si bien le dire les marqueurs commerciaux.

Pour finir, donc, il nous faut prendre un peu de recul, faire preuve d'abstraction. L'informatique est aujourd'hui un roulement à bille de notre production. S'il se grippe, c'est une grande partie de la chaîne qui se gèle avec lui.

Ce mémoire n'achève pas seulement une étude longue d'une année. C'est également l'aboutissement de cinq longues années d'études, où, pratiquement chaque soir et chaque samedi matin, j'ai pu m'investir, infatigable, dans le parcours qui m'a amené jusqu'à ce jour. C'est cet effort qui m'a permis d'achever ce mémoire, tant sur la forme que sur le fond.

Annexe A

Squelette d'une requête DNS

Le document normatif rfc 1035 [13] décrit l'en-tête d'une requête DNS. Celle-ci est en mesure de porter une ou plusieurs interrogations DNS simultanément afin de diminuer le trafic et d'optimiser les réponses. Pour comprendre finement comment fonctionne le dns de bout en bout, il est nécessaire de nous pencher sur cette en-tête :

TAB. A.1 – Composition d'une requête DNS

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ID															
QR	Opcode				AA	TC	RD	RA	Z			RCODE			
QDCOUNT															
ANCOUNT															
NSCOUNT															
ARCOUNT															

Voici la transcription des champs décrits dans le tableau A.1 :

- ID : identifiant de la requête (16 bits). La réponse portera le même identifiant.
- QR : ce champ définit si la requête est une question (0) ou une réponse (1),
- OPCODE : composé de quatre bits, l'opcode spécifie le genre de la requête. Initié lors de l'envoi, il est également présent (copié) dans la réponse. Il peut prendre les valeurs suivantes :

  0 : question standard (QUERY),

  1 : requête inverse (IQUERY),

  2 : requête sur le statut du serveur (STATUS),

  3-15 : réservé pour un usage ultérieur.
- AA : positionné dans les réponses, le champ *Authoritative Answer* spécifie si le serveur contacté est autoritaire sur le domaine présent dans la section *question*.
- TC : *TrunCation* explicite si le message est complet ou s'il a dû être tronqué pour une question de trop grande taille que celle autorisée par le canal de communication.
- RD : ce bit indique au serveur, lorsqu'il est positionné, de continuer la récursion jusqu'à obtenir un résultat.
- RA : le champ *Recursion Available* définit si la récursion est supportée par le serveur interrogé.
- Z : réservé pour un usage futur. Attention, ce code doit obligatoirement être à zéro dans toutes les requettes.

- RCODE : code de retour présent dans la réponse, sur 4 bits. Il peut prendre les valeurs suivantes :
  - **0 :** pas d'erreur, le retour est correct.
  - **1 :** erreur de format, le serveur n'est pas en mesure d'interpréter la demande.
  - **2 :** problème de serveur, ce dernier n'est pas en mesure de traiter les requêtes.
  - **3 :** erreur de nom, le domaine demandé n'existe pas. En général, ce code est utilisé par le serveur autoritaire sur le domaine.
  - **4 :** non implémenté, le serveur ne supporte pas ce type de requête.
  - **5 :** refus de réponse pour des raisons de politique.
  - **6-15 :** Réservé pour un usage futur.
- QDCOUNT : entier de 16 bits non signé spécifiant le nombre d'entrées dans la section question.
- ANCOUNT : entier de 16 bits non signé spécifiant le nombre d'entrées dans la section réponse.
- NSCOUNT : entier de 16 bits non signé spécifiant le nombre de serveurs de nom dans la section autorité d'enregistrement.
- ARCOUNT : entier de 16 bits non signé spécifiant le nombre d'enregistrements dans la section additionnelle.

Annexe B

Configuration des logs de bind

```
logging {

  channel default_file {
file "/var/log/named/default.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};

  channel general_file {
file "/var/log/named/general.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};

  channel database_file {
file "/var/log/named/database.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};

  channel security_file {
file "/var/log/named/security.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};

  channel config_file {
file "/var/log/named/config.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};

  channel resolver_file {
```

```
file "/var/log/named/resolver.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};

  channel xfer-in_file {
file "/var/log/named/xfer-in.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};

  channel xfer-out_file {
file "/var/log/named/xfer-out.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};

  channel notify_file {
file "/var/log/named/notify.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};

  channel client_file {
file "/var/log/named/client.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};

  channel unmatched_file {
file "/var/log/named/unmatched.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};

  channel queries_file {
file "/var/log/named/queries.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};

  channel network_file {
file "/var/log/named/network.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};
```

```
  channel update_file {
file "/var/log/named/update.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};

  channel dispatch_file {
file "/var/log/named/dispatch.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};

  channel dnssec_file {
file "/var/log/named/dnssec.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};

  channel lame-servers_file {
file "/var/log/named/lame-servers.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};


  category default { default_file; };
  category general { general_file; };
  category database { database_file; };
  category security { security_file; };
  category config { config_file; };
  category resolver { resolver_file; };
  category xfer-in { xfer-in_file; };
  category xfer-out { xfer-out_file; };
  category notify { notify_file; };
  category client { client_file; };
  category unmatched { unmatched_file; };
  category queries { queries_file; };
  category network { network_file; };
  category update { update_file; };
  category dispatch { dispatch_file; };
  category dnssec { dnssec_file; };
  category lame-servers { lame-servers_file; };
```

Annexe C

Expression régulière de validation d'une URL

Avec l'aimable autorisation de Vincent Batoufflet [1] :

```
((http\:\/\/(((((((([a-z]|[A-Z])|[0-9])|((([a-z]|[A-Z])|[0-9])(([a-z]|[A-Z])|[0-
9])|\-)*(([a-z]|[A-Z])|[0-9])))\.)*(([a-z]|[A-Z])|(([a-z]|[A-Z])(([a-z]|[A-Z])|
[0-9])|\-)*(([a-z]|[A-Z])|[0-9])))))|((([0-9]+)\.){3}([0-9]+)))(\:([0-9]+))?)(\/(
(((((([a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2}))|[\;\:\@\
&\=])*)(\/((((((a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2})
)|[\;\:\@\&\=])*))*)(\?(((((a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0
-9a-fA-F]{2}))|[\;\:\@\&\=])*))?)?)|(ftp\:\/\/(((((((([a-z]|[A-Z])|[0-9]|[$\-\_\.
\+]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2}))|[\;\?\&\=])*)(\:(((((a-z]|[A-Z])|[0-9]|
[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2}))|[\;\?\&\=])*))?\@)?(((((((a-z]|
[A-Z])|[0-9])|((([a-z]|[A-Z])|[0-9])(([a-z]|[A-Z])|[0-9])|\-)*(([a-z]|[A-Z])|[0
-9])))\.)*(([a-z]|[A-Z])|(([a-z]|[A-Z])(([a-z]|[A-Z])|[0-9])|\-)*(([a-z]|[A-Z])
|[0-9])))))|((([0-9]+)\.){3}([0-9]+)))(\:([0-9]+))?))(\/(((((((a-z]|[A-Z])|[0-9]|
[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2}))|[\?\:\@\&\=])*)(\/((((((a-z]|[A-
Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2}))|[\?\:\@\&\=])*))*)(\;t
ype\=[AIDaid])?)?)|(news\:(\*|(([a-z]|[A-Z])(([a-z]|[A-Z])|[0-9]|[\-\.\+\_])*)|(
(((((a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2}))|[\;\/\?\:
\&\=])+\@((((((([a-z]|[A-Z])|[0-9])|((([a-z]|[A-Z])|[0-9])(([a-z]|[A-Z])|[0-9])|
\-)*(([a-z]|[A-Z])|[0-9])))\.)*(([a-z]|[A-Z])|(([a-z]|[A-Z])(([a-z]|[A-Z])|[0-9
])|\-)*(([a-z]|[A-Z])|[0-9])))))|((([0-9]+)\.){3}([0-9]+))))))|(nntp\:\/\/(((((((
[a-z]|[A-Z])|[0-9])|((([a-z]|[A-Z])|[0-9])(([a-z]|[A-Z])|[0-9])|\-)*(([a-z]|[A-
Z])|[0-9])))\.)*(([a-z]|[A-Z])|(([a-z]|[A-Z])(([a-z]|[A-Z])|[0-9])|\-)*(([a-z]|
[A-Z])|[0-9])))))|((([0-9]+)\.){3}([0-9]+)))(\:([0-9]+))?)\/(([a-z]|[A-Z])(([a-z]
|[A-Z])|[0-9]|[\-\.\+\_])*)(\/([0-9]+))?)|(telnet\:\/\/(((((((([a-z]|[A-Z])|[0-9]
|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2}))|[\;\?\&\=])*)(\:(((((a-z]|[A-Z
])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2}))|[\;\?\&\=])*))?\@)?((((
(((([a-z]|[A-Z])|[0-9])|((([a-z]|[A-Z])|[0-9])(([a-z]|[A-Z])|[0-9])|\-)*(([a-z]|
[A-Z])|[0-9])))\.)*(([a-z]|[A-Z])|(([a-z]|[A-Z])(([a-z]|[A-Z])|[0-9])|\-)*(([a-
```

[1] http ://labs.batoufflet.info/checkurl/regex.txt

```
z]|[A-Z])|[0-9])))|((([0-9]+)\.){3}([0-9]+)))(\:([0-9]+))?))\/?)|(gopher\:\/\/(
((((((([a-z]|[A-Z])|[0-9])|((([a-z]|[A-Z])|[0-9])(([a-z]|[A-Z])|[0-9])|\-)*(([a-
z]|[A-Z])|[0-9])))\.)*(([a-z]|[A-Z])|(([a-z]|[A-Z])(([a-z]|[A-Z])|[0-9])|\-)*((
[a-z]|[A-Z])|[0-9])))))|((([0-9]+)\.){3}([0-9]+)))(\:([0-9]+))?)(\/(((([a-z]|[A-Z
])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|[\;\/\?\:\@\&\=]|(\%[0-9a-fA-F]{2}))((((([a
-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|[\;\/\?\:\@\&\=]|(\%[0-9a-fA-F]{2})
)+)(\%09(((((([a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2})))|
[\;\:\@\&\=])*)(\%09((((([a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|[\;\/\?\:
\@\&\=]|(\%[0-9a-fA-F]{2}))+))?)?)?)?)?)|(wais\:\/\/(((((((([a-z]|[A-Z])|[0-9])|(
(([a-z]|[A-Z])|[0-9])(([a-z]|[A-Z])|[0-9])|\-)*(([a-z]|[A-Z])|[0-9])))\.)*(([a-
z]|[A-Z])|(([a-z]|[A-Z])(([a-z]|[A-Z])|[0-9])|\-)*(([a-z]|[A-Z])|[0-9])))))|((((
0-9]+)\.){3}([0-9]+)))(\:([0-9]+))?)\/(((((a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'
\(\)\,])|(\%[0-9a-fA-F]{2}))*)|wais\:\/\/(((((((([a-z]|[A-Z])|[0-9])|((([a-z]|[A-
Z])|[0-9])(([a-z]|[A-Z])|[0-9])|\-)*(([a-z]|[A-Z])|[0-9])))\.)*(([a-z]|[A-Z])|(
([a-z]|[A-Z])(([a-z]|[A-Z])|[0-9])|\-)*(([a-z]|[A-Z])|[0-9])))))|((((0-9]+)\.){3
}([0-9]+)))(\:([0-9]+))?)\/(((((a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\
%[0-9a-fA-F]{2}))*)\?(((((a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-9
a-fA-F]{2}))|[\;\:\@\&\=])*)|wais\:\/\/(((((((([a-z]|[A-Z])|[0-9])|((([a-z]|[A-Z]
)|[0-9])(([a-z]|[A-Z])|[0-9])|\-)*(([a-z]|[A-Z])|[0-9])))\.)*(([a-z]|[A-Z])|(([
a-z]|[A-Z])(([a-z]|[A-Z])|[0-9])|\-)*(([a-z]|[A-Z])|[0-9])))))|((((0-9]+)\.){3}(
[0-9]+)))(\:([0-9]+))?)\/(((((a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[
0-9a-fA-F]{2}))*)\/(((((a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-9a-f
A-F]{2}))*)\/(((((a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2
}))*))|(mailto\:(((((a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|[\;\/\?\:\@\&
\=]|(\%[0-9a-fA-F]{2}))+))|(file\:\/\/(((((((a-z]|[A-Z])|[0-9])|((([a-z]|[A-Z])
|[0-9])(([a-z]|[A-Z])|[0-9])|\-)*(([a-z]|[A-Z])|[0-9])))\.)*(([a-z]|[A-Z])|(([a
-z]|[A-Z])(([a-z]|[A-Z])|[0-9])|\-)*(([a-z]|[A-Z])|[0-9])))))|((([0-9]+)\.){3}([
0-9]+)))|localhost)\/((((((([a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-
9a-fA-F]{2}))|[\?\:\@\&\=])*)(\/((((((a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\
,])|(\%[0-9a-fA-F]{2}))|[\?\:\@\&\=])*))*))|(prospero\:\/\/(((((((a-z]|[A-Z])|[
0-9])|((([a-z]|[A-Z])|[0-9])(([a-z]|[A-Z])|[0-9])|\-)*(([a-z]|[A-Z])|[0-9])))\.
)*(([a-z]|[A-Z])|(([a-z]|[A-Z])(([a-z]|[A-Z])|[0-9])|\-)*(([a-z]|[A-Z])|[0-9]))
))|((([0-9]+)\.){3}([0-9]+)))(\:([0-9]+))?)\/((((((a-z]|[A-Z])|[0-9]|[$\-\_\.\+
]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2}))|[\?\:\@\&\=])*)(\/((((((a-z]|[A-Z])|[0-9]|
[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2}))|[\?\:\@\&\=])*))*)((\;((((((a-z]
|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2}))|[\?\:\@\&])*)\=(((
([a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2}))|[\?\:\@\&])
*)))*)|((((a-z]|[0-9]|[\+\-\.])+)\:(((((a-z]|[A-Z])|[0-9]|[$\-\_\.\+]|[\!\*\'\(\
)\,])|[\;\/\?\:\@\&\=]|(\%[0-9a-fA-F]{2}))*|(\/\/(((((((a-z]|[A-Z])|[0-9]|[$\-\
_\.\+]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2}))|[\;\?\&\=])*)(\:((((((a-z]|[A-Z])|[0-
9]|[$\-\_\.\+]|[\!\*\'\(\)\,])|(\%[0-9a-fA-F]{2}))|[\;\?\&\=])*))?\@)?(((((((a-
z]|[A-Z])|[0-9])|((((a-z]|[A-Z])|[0-9])(((a-z]|[A-Z])|[0-9])|\-)*(((a-z]|[A-Z])
|[0-9])))\.)*(([a-z]|[A-Z])|(([a-z]|[A-Z])(((a-z]|[A-Z])|[0-9])|\-)*(([a-z]|[A-
Z])|[0-9])))))|((((0-9]+)\.){3}([0-9]+)))(\:([0-9]+))?))(\/(((((a-z]|[A-Z])|[0-9]
|[$\-\_\.\+]|[\!\*\'\(\)\,])|[\;\/\?\:\@\&\=]|(\%[0-9a-fA-F]{2}))*))?))))(?= |$)
```

Annexe D

Code de statut HTTP

```
"100"   ; Continue
"101"   ; Switching Protocols
"200"   ; OK
"201"   ; Created
"202"   ; Accepted
"203"   ; Non-Authoritative Information
"204"   ; No Content
"205"   ; Reset Content
"206"   ; Partial Content
"300"   ; Multiple Choices
"301"   ; Moved Permanently
"302"   ; Moved Temporarily
"303"   ; See Other
"304"   ; Not Modified
"305"   ; Use Proxy
"400"   ; Bad Request
"401"   ; Unauthorized
"402"   ; Payment Required
"403"   ; Forbidden
"404"   ; Not Found
"405"   ; Method Not Allowed
"406"   ; Not Acceptable
"407"   ; Proxy Authentication Required
"408"   ; Request Time-out
"409"   ; Conflict
"410"   ; Gone
"411"   ; Length Required
"412"   ; Precondition Failed
"413"   ; Request Entity Too Large
"414"   ; Request-URI Too Large
"415"   ; Unsupported Media Type
"500"   ; Internal Server Error
"501"   ; Not Implemented
```

```
"502"   ; Bad Gateway
"503"   ; Service Unavailable
"504"   ; Gateway Time-out
"505"   ; HTTP Version not supported
extension-code
```

Annexe E

Fichier de configuration client SSH

Le fichier ssh_config : Ce fichier représente la configuration du client ssh.

```
#$OpenBSD: ssh_config,v 1.15 2002/06/20 20:03:34 stevesk Exp $

# Ceci est le fichier de configuration par défaut des clients ssh du
# système. Voir ssh_config(5) pour de plus amples informations. Ce fichier
# fournit les paramètres par défaut pour les utilisateurs, ces valeurs
# peuvent être changées pour chaque utilisateur dans leur propre fichier
# de configuration ou sur la ligne de commande.

# Les données de configuration sont analysées comme ci-dessous :
# 1. Les options de la ligne de commande
# 2. Le fichier spécifique de l'utilisateur
# 3. Le fichier par défaut
# N'importe quelle valeur de configuration est uniquement changée
# la première fois qu'elle est mise. Ainsi, les définitions spécifiques
# d'hôte doivent être au début du fichier de configuration et les options
#par défaut à la fin.

# Options par défaut
# Pour que les options soient prises en compte, il vous faut enlever
# le dièse précédent la ligne.

# Host permet de spécifier que la configuration qui suit concerne un
# ou plusieurs hôtes précis. Il est possible d'employer des caractères
# joker tels que * ou ?.
# Host *

# Specifie si la connexion à l'agent d'authentification doit être renvoyé
# vers la machine distante.
# ForwardAgent no

# Autorise ou non la redirection du serveur graphique.
```

```
# ForwardX11 no

# Autorise la méthode d'authentification par Rhosts. Peu sûr.
# RhostsAuthentication no

# Autorise la méthode d'authentification par Rhosts sur du RSA. Ne
# fonctionne que dans la version première du protocole et nécessite
# un setuid root. De préférence
# à ne pas utiliser.
# RhostsRSAAuthentication no

# Méthode d'authentification par RSA, ne fonctionne qu'avec la première
# version du protocole. Il faut que le fichier identity existe.
# RSAAuthentication yes

# Autorise la connexion par mot de passe
# PasswordAuthentication yes

# Si l'indicateur est positionné à yes, la demande de passephrase ou de
# mot de passe sera annulée. Cette option est utilisée dans le cas où seul
# des scripts interviennent et où il n'y a pas d'utilisateur présent pour
# insérer le mot de passe.
# BatchMode no

# Vérification de l'adresse IP de l'hôte qui se connecte dans le
# fichier known_hosts

# CheckHostIP yes

# Cette option permet de gérer l'ajout et le changement des clés hôtes
# dans known_hosts.
# Si la clé a changé, la connexion vous sera refusée, vous indiquant
# le motif.

# StrictHostKeyChecking ask

# Localisation des fichiers identity, id_rsa, id_dsa

# IdentityFile ~/.ssh/identity
# IdentityFile ~/.ssh/id_rsa
# IdentityFile ~/.ssh/id_dsa

# Port de connexion à l'hôte distant.

# Port 22

# Version des protocoles supportés et désirés. Ici, on préfèrera employer
```

```
# la deuxième version si elle est disponible.
# Protocol 2,1

# Protocole de chiffrement (ssh v1) et protocoles disponibles par ordre
# de préfèrence (ssh v2).
# Cipher 3des
# Ciphers aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,
# aes256-cbc

# Caractère d'échappement.
# EscapeChar ~
```

Annexe F

Autorisation d'utilisation du schéma de fonctionnement de Syslog-ng

De: Balazs Scheidler <bazsi_balabit.hu>
À: Jean-Philippe Gaulier <jpgaulier_point-libre.org>
Cc: documentation_balabit.com
Sujet: Re: Demand to use a graph of your Syslog-ng documentation
Date: Sat, 26 May 2007 23:07:07 +0200

On Sat, 2007-05-26 at 17:11 +0200, Jean-Philippe Gaulier wrote:
> Hello,
> in order to finish my last degree and graduate as an Engineer, I'm
> writing a report on "How to set up a load-balanced and securised
> architecture based on free software". This one has a chapter on logs,
> and, of course, I'm using the wonderful GPL Syslog-ng.
>
> Nevertheless, I didn't find the licence of your documentation,
> so I'm asking you if it's possible to me to include the following
> picture in my report :
> http://www.balabit.com/dl/html/syslog-ng-admin-guide_en.html/
> logging01.png

The documentation comes with a default "all rights reserved" license,
however hereby I, in the name of BalaBit IT Security Ltd, permit you to
use the above-mentioned picture in your report.

I sign the message with gpg, so you can have a proof that I issued this
permission.

(even though this may or may not be enough in your legislation, it is
certainly more than having an unsigned e-mail, judges usually accept
e-mail as proof)

```
--
Balazs Scheidler
CEO
BalaBit IT Security Ltd.
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.6 (GNU/Linux)

iD8DBQBGWKF7thNE0K3PQTgRAnBaAJ9TJ/n0iFPuNuehKVCQNVANhJ3h0wCaA3ZR
zCtnhSLXqKpSQHa/ry1DqZg=
=++AE
-----END PGP SIGNATURE-----
```

Annexe G

Certificat du site web de l'École Ouverte Francophone

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 2 (0x2)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=FR, ST=France, L=Limoges, O=Ecole Ouverte Francophone,
OU=Eof education, CN=eof.eu.org/emailAddress=info@eof.eu.org
        Validity
            Not Before: Jan 14 14:53:24 2007 GMT
            Not After : Jan 11 14:53:24 2017 GMT
        Subject: C=FR, ST=France, L=Limoges, O=Ecole Ouverte Francophone,
OU=Eof education, CN=*.eof.eu.org/emailAddress=info@eof.eu.org
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:d0:68:1f:e5:22:d8:af:12:9a:c0:00:00:af:e5:
                    be:bd:d2:9c:4e:5a:91:67:4e:bb:96:1a:11:f1:39:
                    34:b7:f2:1c:97:d4:13:7c:f2:6f:67:51:eb:99:6c:
                    97:d8:89:ba:50:c6:bf:99:92:da:e6:96:c7:36:6e:
                    bc:bb:c9:42:2d:7a:d1:49:59:a7:b9:09:50:99:f9:
                    63:bf:c9:20:2b:da:92:db:ef:60:a8:9b:6c:3f:b9:
                    3c:1e:87:72:55:20:69:2e:17:76:80:7e:cf:5a:8d:
                    c8:0c:27:c5:79:8c:db:27:25:f2:1b:11:f8:19:b4:
                    a1:70:4b:e3:d6:8f:a1:c9:53
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
```

```
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                08:BC:24:4A:48:F2:1A:78:9D:CE:FB:99:80:FE:7B
:0E:60:98:6C:99
            X509v3 Authority Key Identifier:
                keyid:EB:B1:5F:E5:B3:06:E1:47:8D:85:9B:53:8C
:4B:90:92:E8:DE:69:9
8
                DirName:/C=FR/ST=France/L=Limoges/O=Ecole Ouverte
Francophone/OU=Eof education/CN=eof.eu.org/
emailAddress=info@eof.eu.org
                serial:E7:E0:4E:9C:E3:00:49:C1

    Signature Algorithm: md5WithRSAEncryption
        4c:ae:94:5e:d3:9e:95:4c:bb:e2:a0:c9:1d:35:06:aa:f5:37:
        60:f6:7a:58:76:45:18:eb:ac:d1:a7:f0:7a:b6:e9:bc:37:14:
        f4:9e:21:a1:bd:f6:5b:94:b0:9b:84:d9:4a:89:56:7e:20:91:
        f0:5e:b2:16:62:82:f6:b0:4a:ad:d8:48:0a:fd:30:71:cc:9c:
        7f:89:3d:bc:f3:f6:11:c0:3c:88:b3:99:23:9f:53:ed:aa:b8:
        c7:e3:ac:ad:0d:3a:03:d1:9e:d6:78:5c:7a:45:86:b3:a9:bb:
        54:f5:e8:e7:0e:96:42:63:65:18:a7:ed:f8:5b:6f:78:b2:c1:
        9c:9a
```

Glossaire

ACL : Access Control List.
AH : Authentication Header.
Anti-spoofing : procédé visant à arrêter tout trafic réseau externe possédant une adresse identique à une machine interne.
ARP : Address Resolution Protocol.
Attaque par dictionnaire : rejeu d'authentification pour un utilisateur donné avec une suite de mots de passe.
Bind : logiciel majeur de gestion des DNS.
Blogs : carnet de bord.
Botnet : réseau de logiciels commandé simultanément par une seule entité.
Buzz : fait divers fortement discuté sur Internet.
CD live : système d'exploitation qui démarre à partir d'un cd-rom.
CERN : Conseil Européen pour la Recherche Nucléaire.
Code source : anatomie d'un logiciel.
Condensat : procédé mathématique appliqué sur des données dont il résulte une chaîne de caractères unique.
Contexte : zone d'exécution.
Cookie : données inscrites par un site Internet sur l'ordinateur à travers un navigateur.
Coupe-feu : voir **Firewall**.
Crontab : ordonnanceur.
Cryptage : procédé irréversible créant un condensat.
CSS : Cascading Style Sheet.
CVS : Concurrent Version System.
DNS : Domain Name System.
DNSsec : Domain Name System SECurity.
DoD : Department of Defense.
E-learning : enseignement à distance.
Empreinte : condensat.
ESP : Encapsulated Security Payload.
Exploit : logiciel permettant d'exploiter une faille de sécurité.
Firewall : équipement ou logiciel mis en coupure d'un réseau afin de le gérer.
FQDN : Fully Qualified Domain Name.
GNU : GNU is Not Unix. Philosophie du logiciel libre.
GPL : GNU Public Licence.
GrSec : patch du kernel Linux visant à accroître la sécurité.
GRUB : chargeur de démarrage GRand Unified Bootloader.
Hash : condensat.
HIDS : Host based Intrusion Detection System.
HMAC : Hashed Message Authentification Control.

Honeynet : réseau constitué de pots de miel.
HOSTS.TXT : fichier permettant la gestion de la correspondance IP/nom avant la naissance du protocole DNS.
HTML : HyperText Markup Language.
HTTP : HyperText Transfer Protocol.
IGC : Infrastructures de Gestion de Clés.
IKE : Internet Key Exchange.
Internet : interconnexion des réseaux.
IP : Internet Protocol.
ipchains : première interface de gestion de Netfilter.
IPFW : pare-feu de BSD.
IPsec : Internet Protocol SECurity.
Iptables : interface de gestion de Netfilter.
IPv4 : (Internet Provotol version 4) version 4 du protocole Internet.
IPv6 : (Internet Protocol version 6) version 6 du protocole Internet.
ISC : Internet Software Consortium.
Kernel land : espace d'exécution utilisateur non privilégié.
Lilo : chargeur de démarrage Linux Loader.
Logiciel libre : logiciel répondant aux 4 libertés de la GPL.
Logs : archive système ou logicielle.
LUG : Linux User Group.
Malware : logiciel malveillant.
Man-in-the-middle : attaque par interception.
MD5 : Message Digest 5.
Mib : (Management Information Base) base de données contenant les informations nécessaires à la gestion d'un système ou d'un parc informatique.
Miredo : implémentation libre de **Teredo**.
Mosaic : premier moteur des navigateurs Internet.
NAT : Network Address Translation.
Netfilter : pare-feu de Linux.
Newsgroups : forum de discussion.
NFS : (Network File System) protocole réseau de système de fichiers.
NORAD : North American Aerospace Defense Command.
NS : Name Server.
Obfuscation : procédé visant à accentuer la difficulté de lecture d'un document.
Open source : logiciel dont le code source est disponible.
OSI : Open System Interconnect.
PACIFICA : technologie AMD de virtualisation matérielle.
Peer-to-peer : protocole d'échange de données selon un mode égalitaire.
PKI : (Public Key Infrastructure) voir **IGC**.
Ready for IPv6 : label symbolisant le fonctionnement avec le protocole IPv6.
Reverse engineering : voir **Rétro-conception**.
RFC : (Request For Comment) ensemble des protocoles réseaux normalisés.
Ring : espace d'exécution système.
Ring0 : espace d'exécution privilégié.
Ring3 : espace d'exécution utilisateur (sans privilège).
Root : administrateur d'un système informatique.
Rot13 : rotation de treize lettres de l'alphabet pour chiffrer un message.
Round-robin : méthode de gestion de file.
Rsync : algorithme et logiciel de synchronisation distante.
Réseaux privés : ensemble de réseaux définis par la RFC 1918.
Rétro-conception : analyse d'un binaire pour en comprendre le fonctionnement.
S/MIME : Secure / Multipurpose Internet Mail Extensions.
Script kiddies : apprenti hacker.

SElinux : patch du kernel Linux développé par la NSA visant à accroître la sécurité.
SHA-1 : Secure Hash Algorithm..
Single mode : mode de démarrage simple sans montage de partition système ni lancement de service.
SNMP : Simple Network Management Protocol.
SOA : (Start Of Authority) début d'une zone DNS qui prendra fin avec la déclaration du prochain SOA.
SSH : (Secure SHell) terminal distant sécurisé.
SSL : (Secure Socket Layer) couche de chiffrement réseau.
Sudo : exécution d'une commande avec les droits d'un autre utilisateur.
Syslog : protocole de centralisation des archives.
Sécurité en profondeur : méthode de sécurisation par niveau.
Tar : algorithme de compression de données.
Tarball : fichier binaire résultant d'un tar.
TCP : Transmission Control Protocol.
Teredo : encapsulation des paquets IPv6 dans des paquets UDP/IPv4.
TLD : Top Level Domain.
Trust : zone interne considérée comme sûre.
TSIG : Transaction Signature.
UDP : User Datagram Protocol.
Unix : UNIplexed Information and Compluting System.
Untrust : zone externe considérée comme non sûre.
URI : Universal Resource Identifiers.
URL : Uniform Resource Locator.
Userland : espace utilisateur.
Vanderpool : technologie Intel de virtualisation matérielle.
VPN : Virtual Private Network.
Wake on lan : technique consistant à allumer un ordinateur à distance via le réseau local.
Watchdog : petit logiciel de surveillance assigné à une tâche précise.
Wiki : zone de travail collaborative.
WWW : World Wide Web.
XML : eXtensible Markup language.

Bibliographie

[1] Olivier Ricou. Apprendre par les logiciels libres. 1re soumission à Environnements Informatiques pour l'apprentissage Humain, Janvier 2007.

[2] Wikipedia. Histoire du cp-40, 2007. http://en.wikipedia.org/wiki/History_of_CP/CMS.

[3] Wikipedia. Machine virtuelle, 2007. http://fr.wikipedia.org/wiki/Machine_virtuelle.

[4] Wikipedia. Virtualisation, 2007. http://fr.wikipedia.org/wiki/Virtualisation.

[5] Wikipedia. Virtualization, 2007. http://en.wikipedia.org/wiki/Virtualization.

[6] Intel. *Intel® Vanderpool Technology for IA-32 Processors (VT-x)*, 2005. http://cache-www.intel.com/cd/00/00/19/76/197666_197666.pdf.

[7] Joanna Rutkowska. Subverting vista kernel for fun and profit. In *Symposium for Security on Asia Network*. Syscan 06, 2006. http://theinvisiblethings.blogspot.com/2006/06/introducing-blue-pill.html.

[8] Brad Spengler. Chroot(), sécurité illusoire ou illusion de sécurité. *Misc 09*, 2003.

[9] Franck Cappello Benjamin Quetier, Vincent Neri. Selecting a virtualization system for grid/p2p large scale emulation. In *Proc of the Workshop on Experimental Grid testbeds for the assessment of large-scale distributed applications and tools (EXPGRID'06), Paris, France, 19-23 june*, 2006. http://www.lri.fr/~quetier/papiers/EXPGRID.pdf.

[10] Wikipedia. Vinton cerf, 2007. http://en.wikipedia.org/wiki/Vinton_Cerf.

[11] Wikipedia. Paul mockapetris, 2007. http://en.wikipedia.org/wiki/Paul_Mockapetris.

[12] Paul Albitz et Cricket Liu. *DNS et BIND*. O'Reilly, 1999.

[13] Paul Mockapetris. Domain names - implementation and specification. RFC 1035, Internet Engineering Task Force, November 1987. ftp://ftp.rfc-editor.org/in-notes/rfc1035.txt.

[14] Wikipedia. Bind, 2007. http://en.wikipedia.org/wiki/BIND.

[15] Wikipedia. Paul vixie, 2007. http://en.wikipedia.org/wiki/Paul_Vixie.

[16] IANA. Special-use ipv4 addresses. RFC 3330, Internet Engineering Task Force, September 2002. ftp://ftp.rfc-editor.org/in-notes/rfc3330.txt.

[17] M. Lottor. Domain administrators operations guide. RFC 1033, Internet Engineering Task Force, November 1987. ftp://ftp.rfc-editor.org/in-notes/rfc1033.txt.

[18] P. Vixie et al. Secret key transaction authentication for dns (tsig). RFC 2845, Internet Engineering Task Force, May 2000. ftp://ftp.rfc-editor.org/in-notes/rfc2845.txt.

[19] Paul Wouters. Dnssec howto. *Secure DNS, Het beveiligingen van DNS in de praktijk*, 2003. http://www.xtdnet.nl/paul/dnssec/.

[20] Jean-Philippe Pick et al. Corrigé des tp dnssec de l'atelier idsa, 2004. http://www.idsa.prd.fr/atelier-idsa/corrige.html.

[21] T. Berners-Lee et al. Uniform resource locators (url). RFC 1738, Internet Engineering Task Force, December 1994. ftp://ftp.rfc-editor.org/in-notes/rfc1738.txt.

[22] T. Berners-Lee et al. Uniform resource identifier (uri) : Generic syntax. RFC 3986, Internet Engineering Task Force, January 2005. ftp://ftp.rfc-editor.org/in-notes/rfc3986.txt.

[23] T. Berners-Lee et al. Hypertext transfer protocol – http/1.1. RFC 2616, Internet Engineering Task Force, June 1999. ftp://ftp.rfc-editor.org/in-notes/rfc2616.txt.

[24] François-Xavier WIPLIER. *Load balancing et clustering sous linux*, 2003. http://www.supinfo-projects.com/fr/2003/load_balancing_et_clustering_sous_linux/.

[25] Benjamin Gigon. *La Haute disponibilité*, 2002. http://lea-linux.org/cached/index/Leapro-pro_sys-dispo.html.

[26] Andrew Tridgell et Paul Mackerras. *The rsync algorithm, Canberra, ACT 0200, Australia*, 1999. http://rsync.samba.org.

[27] Andrew Tridgell. *Efficient Algorithms for Sorting and Synchronization*. PhD thesis, Australian National University, 1999. http://samba.org/~tridge/phd_thesis.pdf.

[28] C. Lonvick. The bsd syslog protocol. RFC 3164, Internet Engineering Task Force, August 2001. ftp://ftp.rfc-editor.org/in-notes/rfc3164.txt.

[29] Dominique De Villepin Premier Ministre. Décret n° 2006-358 du 24 mars 2006 relatif à la conservation des données des communications électroniques. Journal Officiel, March 2006. http://www.legifrance.gouv.fr/WAspad/UnTexteDeJorf?numjo=JUSD0630025D.

[30] C. Adams et al. Internet x.509 public key infrastructure certificate management protocol (cmp). RFC 4210, Internet Engineering Task Force, September 2005. http://www.ietf.org/rfc/rfc4210.txt.

[31] R. Housley et al. Internet x.509 public key infrastructure certificate and crl profile. RFC 2459, Internet Engineering Task Force, January 1999. http://www.ietf.org/rfc/rfc2459.txt.

[32] C Stoll. *The Cuckoo's Egg : Tracking a Spy Through the Maze of Computer Espionage*. Pocket Books, 1990.

[33] Bill Cheswick. *An Evening with Berferd. In Which a Cracker is Lured, Endured, and Studied*. Management Analytics and Others, 1995. http://all.net/books/berferd/berferd.html.

[34] Fuller et al. Classless inter-domain routing (cidr) : an address assignment and aggregation strategy. RFC 1519, Internet Engineering Task Force, September 1993. http://www.ietf.org/rfc/rfc1519.txt.

[35] Rekhter et al. Address allocation for private internets. RFC 1918, Internet Engineering Task Force, February 1996. http://www.ietf.org/rfc/rfc1918.txt.

[36] K. Egevang et P. Francis. The ip network address translator (nat). RFC 1631, Internet Engineering Task Force, May 1994. http://www.ietf.org/rfc/rfc1631.txt.

[37] S. Deering et R. Hinden. Internet protocol, version 6 (ipv6) specification. RFC 2460, Internet Engineering Task Force, December 1998. ftp://ftp.rfc-editor.org/in-notes/rfc2460.txt.

[38] G. Cizault. *Théorie et pratique*. O'Reilly, 2005. http://livre.point6.net/index.php/IPv6.

[39] C. Huitema. Teredo : Tunneling ipv6 over udp through network address translations. RFC 4380, Internet Engineering Task Force, February 2006. http://www.ietf.org/rfc/rfc4380.txt.

[40] P. Biondi A. Ebalard. Ipv6 routing header security. In *CansecWest 2007*. EADS Innovation Works, 2007. http://secdev.org/conf/IPv6_RH_security-csw07.pdf.

Résumé

Cette étude porte sur la mise en place d'une structure informatique type pour un hébergeur à base de logiciels libres, selon un environnement réparti et sécurisé. On y découvre les services les plus utilisés (Web, DNS, Shell…), configurés selon les règles de l'art, mais également des logiciels dédiés à la surveillance et à la consolidation de la plate-forme. Les principaux éclairages portent sur :

- la répartition à travers la virtualisation, la répartition de charge, la copie de données,
- la sécurité en profondeur, en configurant chaque logiciel pour gérer son périmètre ainsi que celui des niveaux supérieurs, mais aussi des environnements dédiées tels qu'un pare-feu ou un pot de miel.

L'évolution technique vers un réseau de type IPv6 est présentée afin d'anticiper les demandes futures.

Mots clés : *logiciel libre*, *sécurité*, *HTTP*, *DNS*, *pare-feu*, *pot de miel*, *virtualisation*.

Summary

This study presents the set up of a web hosting framework based on free software, relying on a secured and shared environment. Best practices for configuration of most current services (Web, DNS, Shell…) are studied. Then dedicated softwares to supervise and consolidate the platform are presented. Focus is on :

- balancing through virtualization, loadblancing, data replication,
- in-depth security, achieved through configuration of each software to manage its own perimeter as well as upper layers, and dedicated environments such as firewall or honeypot.

The technical evolution to an IPv6-enabled network is discussed in order to plan further requirements.

Key words : *free software*, *security*, *HTTP*, *DNS*, *firewall*, *honeypot*, *virtualization*.