Mettre en application une solution coupe-feu/VPN souple et modulable à moindre coût

Nino Margetic, Alain Astgen

Mettre en application une solution pare-feu/VPN souple et modulable à moindre coût

Nino Margetic
Centre National de Genotypage
2 rue Gaston Cremieux, CP5721, 91057 Evry CEDEX
nino.margetic@cng.fr

Date: 15 octobre 2003

Alain Astgen
Centre National de Genotypage
2 rue Gaston Cremieux, CP5721, 91057 Evry CEDEX
alain.astgen@cng.fr

Résumé

De nos jours, une connexion à haut débit est relativement peu onéreuse et beaucoup de gens ont la possibilité de travailler à partir de leur domicile. Toutefois, il existe des inconvénients liés à une connexion permanente à l'internet. Compte tenu de nombreux avertissements de sécurité, il existe un besoin réel de mise en place d'une solution de pare-feu et Réseau Privé Virtuel (VPN), qui soit souple, à moindre coût et à maintenance réduite. Nous utilisons, le système d'exploitation gratuit OpenBSD, comme pare-feu et solution VPN. Le développement d'OpenBSD a commencé il y a sept ans, grâce à l'audit d'un code source existant. Dès le début, les efforts ont porté sur la sécurité proactive et la cryptographie intégrée. L'inclusion du démon ISAKMP lui-même, sans même faire mention que le système OpenBSD est utilisé comme plate-forme de référence dans le contrôle de l'interopérabilité VPN, fait de lui un candidat idéal pour la mise en place d'une solution VPN à moindre coût et à gestion réduite. Quoique d'un côté, ses exigences limitées lui permettent de fonctionner sur un matériel totalement obsolète, OpenBSD supporte tout aussi bien des connexions gigabit multiples, supportant de ce fait tout aussi bien les solutions pare-feu/VPN de type entreprise. Notre pratique s'étend du pare-feu simple à double interface sur réseau privé via l'ADSL aux configurations multi port au niveau de l'entreprise. Nous avons développé des configurations VPN d'OpenBSD qui supportent tout aussi bien le type de connexions VPN statique que le type « roadwarrior » et un certain nombre de passerelles de sécurité différentes tels qu'OpenBSD, Cisco et Windows. Nous utilisons le système OpenBSD comme passerelle pare-feu/VPN sur des systèmes de réseaux variés, en passant par les lignes ISDN et DSL jusqu'aux réseaux Ethernet multi-gigabit et nous l'avons trouvé extrêmement performant, résistant, facile à mettre en place et à utiliser, et sans grande exigence de maintenance.

Mots clefs

Sécurité, OpenBSD, pare-feu ou firewall, Réseaux Privés Virtuels

1 Introduction

Il y a peu de temps encore, faire fonctionner un système informatique sécurisé était chose aisée : il suffisait de débrancher toutes les connexions via modem et d'autoriser uniquement les terminaux directement connectés, d'installer la machine et ses terminaux dans une pièce protégée, et de poster un garde à l'entrée [1]. Pour le meilleur ou pour le pire, la majorité des systèmes informatiques ne fonctionnent pas de cette manière de nos jours ; ils sont installés dans un bureau type ou à domicile avec une connexion à l'internet. Avec l'adoption rapide de l'ADSL et de technologies similaires, la connexion rapide à l'internet fait désormais partie de chaque foyer. Malgré ces changements radicaux, le concept de la sécurité informatique ne s'est pas modifié, empêchant quiconque de faire n'importe quoi avec, sur ou à partir des ordinateurs ou autres périphériques. La politique de sécurité, un ensemble de décisions qui détermine collectivement la position d'un organisme vis-à-vis de la sécurité, définit les limites d'un comportement acceptable et ce que devrait être la réponse aux infractions.

Les systèmes informatiques sécurisés doivent faire face à deux défis : premièrement, utiliser une technologie sophistiquée qui est difficile à concevoir et à s'avérer juste ; deuxièmement, avoir une utilisation aisée pour tout un chacun. On néglige parfois la question de facilité d'utilisation mais elle est essentielle ; une sécurité faible mais facile d'utilisation peut se révéler plus efficace qu'une sécurité forte mais difficile d'utilisation s'il est plus probable que celle-ci soit utilisée. Etant donné que les environnements se développent et se diversifient de plus en plus et que sécuriser ces derniers sur une base hôte par hôte devient de plus en plus délicate (des incidents récents avec le ver Blaster sur MS-Windows illustrent ce point de façon frappante), l'informatique s'est progressivement éloignée du typique « pas de sécurité », « sécurité via l'obscurité »

et des modèles de sécurité basés sur l'hôte pour s'acheminer vers la sécurité réseau. Le modèle de sécurité réseau est axé sur le contrôle de l'accès au réseau à des hôtes différents et aux services qu'ils offrent, plutôt que sur leur sécurité individuelle. Les techniques de sécurité réseau comprennent l'intégration de pare-feux (« firewalls ») pour la protection des systèmes et réseaux internes grâce à l'utilisation de techniques d'authentification puissantes (telles que les mots de passe à usage unique) et de chiffrement pour la protection de données particulièrement sensibles puisqu'elles passent par le réseau. Un site sera dépendant de ses efforts en matière de sécurité, comme par exemple l'utilisation d'un modèle de sécurité réseau. Par exemple, un simple pare-feu tel que celui qui est examiné ici, protègera facilement, soit une machine unique, soit des milliers de machines contre une attaque de réseaux au-delà du firewall, indépendamment du niveau de sécurité des machines individuelles.

La question en tout premier lieu - pourquoi avoir choisi le système OpenBSD [2] et non pas Linux pour la mise en œuvre de l'infrastructure de sécurité ? - peut s'argumenter de la manière suivante [3, 4]. Bien qu'il y ait un assez grand nombre de solutions à la fois gratuites et/ou commerciales basées sur le système Linux présentant des caractéristiques de sécurité introuvables sur le système OpenBSD avec des distributions Linux complètes [5, 6], des corrections de noyaux [7], des progiciels supplémentaires [5], toujours est-il qu'avec le système Linux, la sécurité vient après coup. Le système OpenBSD a été conçu dès le début en gardant à l'esprit l'idée de sécurité.

Le système OpenBSD contient OpenSSL[8], OpenSSH [9], IPSec et plusieurs autres logiciels cryptographiques (l'équipe de l'OpenBSD a été en grande partie chargée des développements d'OpenSSH). Alors que de nombreux vendeurs Linux proposent effectivement OpenSSH et OpenSSL, il y a seulement quelques vendeurs majeurs intégrant IPsec. Bien qu'il existe un projet pour Linux IPSec [10], il s'avère difficile au mieux de l'installer et de le configurer. Etant donné que la majorité du travail sur Linux est réalisée à partir des Etats-Unis, il n'existe presque pas de support cryptographique intégré au noyau (en raison des restrictions à l'exportation). Si l'on souhaite ajouter la cryptographie, on doit corriger le noyau et le reconstruire. Très peu de vendeurs Linux proposent une cryptographie quelconque intégrée au noyau telle que IPSec, ou une quelconque forme de connexions cryptographiques (cependant, ils sont nombreux à proposer OpenSSL/OpenSSH et d'autres éléments cryptographiques). Le système OpenBSD, par ailleurs, intègre par défaut l'une des meilleures mises en application IPSec disponibles et fournit un démon pour l'échange des clefs, avec le support nécessaire aux différentes formes d'authentification.

Le matériel cryptographique est encore un autre domaine dans lequel OpenBSD brille et dans lequel Linux fait presque complètement défaut. Le système OpenBSD supporte plusieurs produits d'accélération cryptographiques [11-14], permettant, par exemple, de construire des passerelles IPSec très puissantes (et à moindre coût). Quoiqu'il existe certains matériels d'accélération SSL à la disposition de Linux, c'est avant tout un problème facile à résoudre (la plupart des gestionnaires de charge des serveurs web peuvent traiter le chiffrement, et conserver une organisation correcte des sessions). Il existe très peu de produits matériels capables d'accélérer IPsec pour Linux. OpenBSD travaille aussi actuellement en vue de permettre au matériel d'accélérer d'autres logiciels cryptographiques tels que SSH, ce qui deviendra un problème de plus en plus important (combien de CPU aurait-on à ajouter à un serveur afin de supporter 1000 utilisateurs employant SSH à la place de Telnet ?). De plus, avec le support de l'OpenSSH pour d'importants transferts de fichiers (via scp et sftp), la charge de ces serveurs utilisant le protocole SSH ne fera qu'augmenter.

2 Matériels et Méthodes

2.1 Le système OpenBSD

Le projet sur l'OpenBSD remonte à octobre 1995 lorsqu'une équipe de développeurs a développé du code à partir de NetBSD et a lancé un nouveau projet autour d'un système d'exploitation similaire à Unix, disponible gratuitement, basé sur 4.4BSD muti-plateforme en portant l'accent sur l'exactitude du code, la sécurité, la standardisation et la portabilité. Depuis le tout début, les idées de la première licence BSD ont été étroitement suivies, aboutissant au fait qu'aujourd'hui toutes les parties du système OpenBSD ont suffisamment de termes de copyright, qu'on autorise une redistribution gratuite et la capacité de réutilisation de la majorité des parties de l'arborescence source, à des fins personnelles ou commerciales.

A l'heure actuelle, la caractéristique probablement la plus évidente d'OpenBSD, comparativement à d'autres systèmes d'exploitation, est son orientation vers une sécurité forte. L'aspiration initiale, pour la prise de position de leader dans l'industrie de la sécurité, ajoutée au modèle de développement de logiciels ouverts, a permis aux développeurs d'avoir une vision sans concession vers une sécurité accrue en comparaison avec d'autres vendeurs. Plus important encore, puisque le système OpenBSD est exporté avec la cryptographie intégrée, les développeurs ont été en mesure de prendre des méthodes cryptographiques en vue de résoudre des problèmes de sécurité.

Plus significativement, l'équipe d'audit de sécurité du projet (habituellement constituée de six à douze membres) qui audite de manière proactive le code source d'OpenBSD depuis l'été 1996, continue à chercher et à réparer de nouvelles failles de sécurité. Ce processus est basé sur l'analyse globale fichier par fichier de chaque élément essentiel du logiciel, pas tant en cherchant les failles de sécurité, qu'en cherchant les bogues élémentaires du logiciel. On a trouvé des défauts dans quasiment chaque zone du système. Au cours de l'audit, de nouveaux types complets de problèmes de sécurité ont été découverts, et le code source qui avait été audité précédemment nécessite souvent un nouvel audit en ayant à l'esprit ces nouveaux défauts. Le code est audité plusieurs fois, et par de nombreuses personnes disposant de compétences différentes en matière d'audit. Enfin, une fois les défauts découverts, ils sont publiés sur le site internet d'OpenBSD sous la forme d'une page web Erratum et d'une correction qui résout un problème spécifique.

Débutant par la version 2.7, l'équipe de l'OpenBSD a commencé à fournir une arborescence source contenant des corrections et des réparations importantes (c'est-à-dire celles qui sont indiquées dans la page Erratum ainsi que d'autres qui étaient évidentes et simples, mais qui ne méritaient pas d'entrée Erratum). Ces corrections ont été mises à disposition via CVS en sus de l'arborescence source actuelle. Ainsi les utilisateurs peuvent désormais choisir trois options :

- S'en tenir à la dernière publication officielle et appliquer les corrections (« patches ») à la main.
- Utiliser le code source de dépôt de CVS dénommé « stable patch branch » avec les corrections appliquées les plus importantes.
- Utiliser le code source de dépôt de CVS dénommée « current patch branch » pour l'ensemble des derniers correctifs.

La page web du « Daily Changelog » sur le site web d'OpenBSD décrit les corrections postérieures qui ont été intégrées au « stable patch branch ». Le principe général veut que toutes les entrées Errata fusionnent dans le « stable patch branch » dans les 48 heures, à compter de la publication d'un Erratum. D'autres corrections postérieures peuvent être également intégrées, sous un certain nombre de conditions :

- Les corrections doivent être simples, courtes, et évidemment correctes à 100 %.
- Les entrées Errata sont établies pour les bogues concernant de nombreuses personnes. D'autres corrections peuvent être jointes au « stable patch branch » si elles touchent quelques personnes de manière très importante.
- De grands sous-systèmes ou d'importantes corrections ne sont jamais jointes au « stable patch branch ».
- En tant qu'exception aux règles ci-dessus, les dernières versions d'OpenSSH sont habituellement intégrées au « stable patch branch ».

Etant donné qu'il n'existe pas de versions binaires de l'OpenBSD (autres que les versions officielles), la procédure exacte de correction du système est la suivante : après l'obtention des dernières corrections, il est nécessaire de reconstruire les binaires concernés (dans le cas d'un « point patch »), ou bien de reconstruire le noyau (dans le cas d'un problème plus complexe), suivi d'une réinitialisation avec le nouveau noyau et ensuite de procéder à la reconstruction des binaires concernés du système.

2.2 Filtre de paquets (PF)

Le filtrage de paquets est peut-être l'aspect le plus important de la sécurité informatique proactive et efficace. Depuis 1996, le filtrage de paquets IPFilter [15] de Darren Reed a été intégré à l'installation de base d'OpenBSD. Concernant la version 3.0 d'OpenBSD, IPFilter a été supprimé du système initial (en raison d'un changement de licence) et l'occasion a été saisie pour développer un nouveau filtre de paquets [16] qui emploie des structures de données et des algorithmes optimisés pour réaliser une bonne performance pour le filtrage de paquets et la translation d'adresses. L'évaluation de l'ensemble des règles se classifie avec O(n), si *n* correspond au nombre de règles dans l'ensemble. Toutefois, la consultation d'état se classifie avec O(log m), si *m* correspond au nombre d'états.

Un filtre de paquets contrôle les paquets atteignant l'une de ses interfaces et prend une décision basée sur des règles en fonction desquelles il acheminera ou n'acheminera pas ceux-ci via une autre interface jusqu'à destination. En raison du processus IP et des protocoles contenus, il est impossible de contrôler un paquet unique au milieu de la transaction et de définir si oui ou non il est valable ; un « *Stateful Packet Filter* » vérifiera les paquets dans le contexte global des connexions déjà établies. Le filtre de paquets PF se trouve dans le noyau et contrôle chaque paquet IP entrant ou quittant la mémoire. Ceci peut conduire à une décision parmi plusieurs : transmettre un paquet non modifié ou modifié, bloquer en silence un paquet ou le rejeter accompagné d'une réponse (par ex : envoi d'une réinitialisation TCP). Le filtre lui-même est composé de deux éléments de base : des règles de filtrage et la table d'état.

Chaque paquet est comparé à l'ensemble des règles de filtrage. L'ensemble des règles se compose d'une liste associée de règles. Chaque règle contient un ensemble de paramètres définissant l'ensemble de paquets pour lesquels la règle s'applique. Les paramètres peuvent être les suivants : l'adresse source ou l'adresse destination, le protocole, le numéro de port, etc. Pour un paquet correspondant à la règle, on réalise la transmission ou le blocage de ce dernier. L'action de bloquer signifie que le paquet est supprimé par le filtre alors que l'action de transmettre veut dire que le paquet est acheminé à destination. Au cours de l'évaluation de l'ensemble des règles, le paquet est comparé à toutes les règles du début à la fin. Un paquet peut correspondre à plus d'une règle, auquel cas la dernière règle correspondante est utilisée. Ce processus permet de passer outre les règles générales pour aller vers des règles plus spécifiques, telles qu'en premier lieu le blocage de tous les paquets et, par la suite, la transmission de paquets spécifiques. La dernière règle correspondante détermine si le paquet sera transmis ou bloqué en fonction de son domaine d'action. Une règle correspondante qui a été signalée en dernier lieu met fin à l'évaluation par les règles du paquet courant et l'action associée à cette règle est appliquée au paquet. Ceci empêche que les dernières règles soient annulées par les règles suivantes.

Le filtrage de paquets avec état (« stateful ») implique qu'un pare-feu contrôle non seulement les paquets uniques, mais aussi qu'il ait connaissance des connexions établies. Toute règle transmettant un paquet peut créer une entrée dans la table d'état. Avant que l'ensemble des règles de filtrage ne soit évalué pour un paquet, on recherche dans la table d'état pour une entrée correspondante. Si un paquet fait partie d'une connexion tracée, il est transmis sans conditions, sans évaluation par l'ensemble des règles. Pour le TCP, la correspondance d'état implique le contrôle des nombres de séquence par rapport aux fenêtres escomptées [17], ce qui améliore la sécurité contre les attaques de nombres de séquence. L'UDP est sans état (« stateless ») par nature, on considère que les paquets font partie de la même connexion, si les adresses et les ports des hôtes correspondent à une entrée de la table d'état. Les entrées d'état UDP ont un schéma basé sur un temps d'expiration. Le premier paquet UDP pourrait être, soit un paquet individuel, soit le commencement d'une pseudo-connexion UDP. Le premier paquet créera une entrée d'état avec un court temps mort. Si l'autre extrémité répond, le filtre de paquets jugera que c'est une pseudo-connexion à communication bidirectionnelle et il permettra plus de souplesse dans la durée de l'entrée d'état. Les paquets ICMP se retrouvent dans deux catégories : les messages d'erreur ICMP qui font référence à d'autres paquets et les interrogations et les réponses ICMP qui sont traitées séparément. Si un message d'erreur ICMP correspond à une connexion dans une table d'état, il y a transmission. Les interrogations et les réponses ICMP créent leur propre état, identique aux états UDP. Prenons un exemple : une réponse écho ICMP correspond à l'état qu'une requête écho ICMP créée. Ceci est nécessaire et ce, afin que les applications telles que le ping ou le traceroute fonctionnent à travers le PF.

En plus du filtrage de paquets, le PF applique également la translation d'adresse et redirection (NAT ou « *Network Address Translation »)*, la normalisation des paquets, la modulation des nombres de séquence et la traçabilité. NAT est couramment utilisée afin de permettre aux hôtes disposant d'adresses IP à partir d'un réseau privé, de partager une connexion Internet grâce à l'utilisation d'une adresse unique publique. Une passerelle NAT remplace les adresses IP et les ports du paquet qui traversent la passerelle par ses propres informations d'adresses. Différents systèmes d'exploitation résolvent le chevauchement de fragments IP de diverses façons. Certains conservent les anciennes données alors que d'autres remplacent les anciennes données par des données issues d'un fragment nouvellement arrivé. Pour les systèmes qui résolvent les chevauchements en faveur de nouveaux fragments, il est possible de ré-écrire (normaliser) les en-têtes de protocole après un contrôle par le pare-feu. Pour les hôtes disposant de générateurs de nombres de séquence initiaux faibles, le PF peut protéger des attaques de nombres de séquence en modulant les nombres de séquence TCP en ajoutant un nombre aléatoire à l'ensemble des nombres de séquence d'une connexion, éliminant de ce fait la possibilité de parodier une conversation entre des hôtes. Une traçabilité étendue est disponible (en temps réel également) en format binaire (tcpdump) via un « pseudo-device ».

Afin de configurer un pare-feu de filtrage de paquets basé sur l'OpenBSD pf(4), l'étape la plus importante est la préparation d'une configuration de fichier pf.conf(5) du PF qui soit conforme à la politique de sécurité reconnue. Le PF lit ses règles de configuration à partir de pf.conf(5) au moment de l'initialisation, alors qu'elles sont chargées par les scripts rc (en modifiant le rc.conf(8) et en initialisant pf=YES). On peut constater qu'alors que le pf.conf(5) est le défaut et qu'il est chargé par les scripts rc du système, cela correspond simplement à un fichier de textes chargé et interprété par pfctl(8) et inséré dans pf(4). Pour certaines applications, il est possible de charger d'autres ensembles de règles à partir d'autres fichiers après l'initialisation. Afin d'activer immédiatement le PF, on peut, soit réinitialiser le système, soit exécuter le programme associé pfctl(8) avec l'option –e. Notez bien que ceci valide simplement le PF mais ne charge pas l'ensemble des règles en fait. Les règles doivent être chargées séparément, soit avant, soit après validation du PF.

Le fichier pf.conf(5) se compose de sept parties : les macros (variables définies par l'utilisateur pouvant contenir les adresses IP, les noms d'interfaces, etc) ; les tables (structures utilisées pour contenir des listes d'adresses IP) ; les options (différentes options de contrôle du fonctionnement du PF) ; le « scrub » (retraitement des paquets pour leur normalisation et

leur défragmentation) ; les files d'attente (fournissant un contrôle des largeurs de bandes et l'organisation des priorités des paquets) ; la translation (contrôles NAT et réorientation des paquets) et les règles de filtrage (permettant un filtrage ou un blocage sélectif des paquets lorsqu'ils traversent n'importe quelle interface). A l'exception des macros et des tables, chaque section devrait apparaître dans l'ordre mentionné au sein du fichier de configuration, bien qu'il ne soit pas nécessaire que toutes les sections existent pour chaque application spécifique. On ignore les lignes en blanc, et les lignes débutant par # sont traitées comme des commentaires.

2.3 Réseau Privé Virtuel (VPN)

Fergusson et Houston [18, 19] ont proposé une définition quelque peu formelle du VPN : environnement de communications dans lequel l'accès est contrôlé afin de permettre les connexions des pairs uniquement au sein d'une communauté d'intérêt définie ; il est construit sous une certaine forme de partage d'un moyen de communication sous-jacent commun, dans lequel ledit moyen de communication sous-jacent fournit des services au réseau sur une base non-exclusive. Toutefois, ils ont également fourni une description plus simple, plus proche et beaucoup moins formelle : un VPN est un réseau privé construit au sein d'une infrastructure de réseau public, tel que l'Internet global.

Dans cette partie, nous considérerons le réseau public comme étant Internet. Dans ce cas précis, les définitions mentionnées ci-dessus ne contiennent pas un concept clé : le « tunneling ». Les solutions VPN, ce qui est le cas très souvent, établissent des tunnels pour traiter l'Internet comme un saut entre deux parties amicales. Les extrémités d'un tunnel encapsulent un paquet de données dans un autre (en utilisant si possible un protocole différent). Le « tunneling » est une technique puissante qui est employée dans de nombreux domaines, par exemple pour le routage des paquets d'un protocole via un réseau utilisant un autre nomade IP ou pour ledit nomade IP. Voici une liste des propriétés des VPN sur lesquelles s'axer :

- Le VPN utilise l'Internet comme infrastructure publique de communications.
- Les VPN garantissent la vie privée au niveau du réseau. Autrement dit, celle-ci est assurée par paquet. Le moyen technique pour y parvenir est d'encapsuler les paquets IP dans d'autres paquets IP (« tunneling ») et d'utiliser des mécanismes cryptographiques pour l'authentification et le chiffrement du contenu.
- Les VPN partagent l'Internet en permettant l'utilisation interne de schémas d'adressage privé (par ex. pour les sites Intranet).

2.4 Une technologie de validation des VPN : l'IPsec

Les techniques traditionnelles de sécurité systèmes se sont axées sur des solutions dépendantes d'applications de haut niveau. Indépendamment de cela, la sécurité de la couche réseau a été admise comme un élément nécessaire dans l'architecture de sécurité multicouche. Au début des années 90, différents membres de la *Internet Engineering Task Force* (IETF) ont décidé de concevoir et d'utiliser un protocole adapté à l'environnement Internet à grande échelle, qui utilise l'IP comme protocole de réseau. Bien que cette première mise en application [20] n'ait jamais tout à fait fonctionné, ceci a montré que le concept était solide et qu'il était possible d'atteindre les objectifs. Quelques années après, Le Groupe de Travail sur l'IPsec de l'IETF a développé un ensemble de spécifications et s'en est suivie en 1995 une première session d'interopérabilité entre les mises en applications d'IPsec de plusieurs vendeurs et des particuliers. Un certain nombre de RFC (RFC ou *Request For Comments* : série de documents représentant un ensemble de notes techniques et organisationnelles sur l'Internet) et d'*Internet Drafts* décrivent l'architecture IPsec [21] comme un ensemble de protocoles qui fournissent l'intégrité des données, la confidentialité, la protection de la rediffusion et l'authentification au niveau des couches réseaux. Ce positionnement au niveau du réseau offre une souplesse considérable grâce à une utilisation manifeste d'IPsec dans différents rôles (par ex : intégration de VPN ou Réseaux Privés Virtuels, protection bout à bout, accès à distance, etc.). Il est impossible d'obtenir une telle souplesse dans des niveaux d'abstraction plus élevés ou plus bas.

L'architecture globale de l'IPsec se compose de trois modules [22]:

- Le protocole de chiffrement des données (ESP) [23] et les protocoles d'authentification (AH) [24]. Ce sont les « wire protocols » utilisés pour encapsuler les paquets IP à protéger. Les paquets sortants sont authentifiés, codés et encapsulés juste avant leur envoi sur le réseau tandis que les paquets entrants sont désencapsulés, vérifiés et déchiffrés dès réception. Ces protocoles sont habituellement mis en application à l'intérieur du noyau pour des raisons de performance et de sécurité.
- Le protocole d'échange des clefs IKE [25] est utilisé pour créer et conserver de façon dynamique les SA ou *Security Associations*. Une SA définit un ensemble de paramètres nécessaires à la communication sécurisée simple entre deux hôtes (par ex : clés cryptographiques, choix des algorithmes, ordre des transformations, etc). Bien que les « wire

protocols » puissent être utilisés individuellement grâce à une gestion de clés manuelle, un développement considérable et une large utilisation d'IPsec sur l'Internet exigent la création automatisée d'une SA sur demande. En raison d'un grand nombre et d'une grande variété de configurations et d'options qu'une mise en œuvre spécifique de l'IKE doit supporter, cette partie de l'architecture d'IPsec tend à dominer les deux autres en termes de taille de code et de complexité.

- Le module de politique (« policy module ») régit le traitement des paquets en route au sein de et en dehors d'un hôte conforme à IPsec. Bien que les protocoles de sécurité protègent du trafic les données, ils ne soulèvent pas la question de savoir : quel hôte est autorisé à échanger quel type de trafic avec l'autre hôte. Bien que des mécanismes de filtrage de paquets traditionnels tels que ceux qui sont utilisés dans les pare-feux modernes, puissent être employés (avec des modifications mineures) pour appliquer des politiques sur le trafic, il est nécessaire d'avoir un mécanisme de niveau supérieur pour la validation et la configuration de tels filtres.

La mise en application d'IPsec dans le système OpenBSD est intégrée au noyau et désigne seulement d'autres protocoles de transport IP (AH et ESP). Le paquet entrant IPsec destiné à l'hôte local est soumis au protocole IPsec approprié pour traitement, sur la base d'un numéro de protocole dans l'en-tête IP. Avec l'utilisation des informations provenant directement du paquet, la SA nécessaire au traitement est localisée dans une version modifiée de la table de routage (noyau) (SPD ou *Security Policy Database*). Une fois le paquet correctement traité (déchiffré, authentifié, etc.), il est ré-acheminé pour un traitement additionnel par le module IP, accompagné d'informations complémentaires (telles que le fait qu'il ait été reçu de manière sécurisée) pour être utilisé par des protocoles supérieurs et au niveau des sockets. Les paquets sortants exigent quelque peu un traitement différent : lorsqu'un paquet est remis au module IP pour transmission, une consultation est réalisée dans la SPD afin de déterminer si le paquet nécessite un traitement par IPsec. Si c'est le cas, la consultation retourne la(les) SA à utiliser pour le traitement IPsec du paquet. Une fois traité, le paquet est retourné à l'IP pour transmission. S'il n'y a pas de SA couramment créée avec l'hôte récepteur, le paquet est supprimé et un message est envoyé au démon de gestion de clés qui devra renégocier les SA nécessaires.

Comme il est indiqué précédemment, afin de développer IPsec sur une grande échelle, il est nécessaire d'avoir une méthode automatisée de gestion des SA et de gestion des clés. Quelques unes des questions rencontrées sont les suivantes : négociation des attributs des SA, authentification, distribution sécurisée des clés et « vieillissement » des clés. Le protocole recommandé par l'IETF est désigné sous le nom d'IKE ou *Internet Key Exchange* ; il est basé sur un protocole de cadre de travail appelé ISAKMP, mettant en oeuvre l'échange de clés Oakley [26]. Le protocole IKE se compose de deux phases : la première phase établit un canal sécurisé entre deux démons de gestion de clés, tandis que dans la deuxième phase, il est possible de négocier directement les SA d'IPsec. La première phase négocie au moins une méthode d'authentification, un algorithme de chiffrement, un algorithme « hash » et un groupe Diffie-Hellman [27]. Cet ensemble de paramètres est dénommé « Phase 1 SA ». Avec l'utilisation de ces informations, les pairs s'authentifient et calculent le matériel clé à utiliser pour la protection de la Phase 2. En fonction de la suite de la protection spécifiée au cours de la Phase 1, on peut utiliser différents modes afin d'établir une Phase 1 SA, les deux plus importants étant le « mode principal » et le « mode agressif ». Le mode principal (« main mode ») fournit une protection d'identité en transmettant les identités des pairs de manière codée. Le mode agressif (« aggressive mode ») offre des garanties plus limitées mais exige moins de messages et tient compte des types de configuration « nomades » en utilisant une authentification basée sur des partages de secrets. La seconde phase est couramment appelée le « mode rapide » et entraîne un « n-uplet » IPsec SA (un entrant et un sortant). Comme le mode rapide (« quick mode ») est protégé par une Phase 1 SA, il n'est pas nécessaire qu'il fournisse sa propre protection d'authentification, ce qui explique une négociation rapide (d'où le nom). On peut choisir d'effectuer un nouveau calcul Diffie-Hellman sous la condition du PFS ou *Perfect Forward Secrecy*. Le PFS est un attribut de communication codé qui tient compte d'une clé de session transitoire pour une transaction obtenue sans affecter la sécurité de futures clés négociées sous la même Phase 1 SA (en d'autres termes, l'ensemble des clés de session sont autonomes d'un point de vue cryptographique). Par conséquent, étant donné les principes décrits du fonctionnement de l'ISAKMP (admission des paquets entrants, réalisation de quelques traitements dans le contexte prescrit du paquet, envoi d'une réponse), les développeurs du système OpenBSD ont réalisé une application basée sur les messages.

La configuration du VPN sur le système OpenBSD est un processus complexe, du fait que l'IPsec et l'IKE sont des protocoles compliqués. Afin de configurer le démon ISAKMP, il est nécessaire de préparer un fichier isakmpd.conf(5) dans une syntaxe plutôt générique identique à MS-Windows. Le format de fichier « .INI » est probablement plus adapté à la machine et aux protocoles, puis aux opérateurs (pour les exemples, se reporter à l'annexe). En interne, tout est traité en triplet (section, étiquette, valeur), les valeurs pouvant de façon optionnelle correspondre aux listes de valeurs d'échelle. Les valeurs elles-mêmes sont souvent des noms de section, donnant de ce fait une structure d'arborescence (ou plutôt une « forêt ») aux données. Une des particularités les plus utiles du démon ISAKMP est sa capacité à fonctionner au premier plan (pour des raisons de débogage) en envoyant des messages de notation (« logging ») à l'erreur type. Le module de

« logging » dispose d'un mécanisme de contrôle à grain fin qui facilite le choix des informations détaillées sur certains sujets. Afin de faciliter la localisation du problème, il est possible d'entrer pratiquement chaque calcul intermédiaire.

3 Résultats

Ce qui a essentiellement motivé l'écriture de cet article était le manque de documentation détaillée sur la configuration et le fonctionnement des pare-feux et des passerelles VPN sur le système OpenBSD. Il est possible d'obtenir des articles théoriques sur le sujet ou bien des documents de type FAQ ou *Frequently Asked Questions* qui donnent des recettes pratiques pour la mise en application d'une solution dans la réalité. Cet article tente de combler le fossé qui existe entre les deux mondes et expose les grandes lignes de la théorie de base avec les détails de la mise en application (dans la réalité), ce qui devrait fournir un bon point de départ. Grâce à une expérience accumulée sur 10 années d'utilisation active d'OpenBSD, nous présenterons les trois cas typiques d'un scénario pare-feu et passerelle VPN que nous avons rencontrés dans notre pratique jusqu'à ce jour. Tout en gardant cela à l'esprit, nous avons décidé de fournir la configuration complète de la machine pour les trois cas de scénario dans la partie « annexe » de l'article et dans ce paragraphe, nous décrirons la disposition générale du réseau pour chacun des cas.

Le Centre National de Génotypage (CNG) est un organisme qui compte environ 100 utilisateurs en informatique, connectés à l'Internet via deux liaisons Ethernet d'1Gbps au REVE (Réseau d'Evry Val d'Essonne) [28], un câble MAN ou *Metropolitan Area Network* de 18 Kms (72 fibres) utilisant BGP [29] comme protocole de routage. On emploie une des liaisons comme point d'entrée et de sortie de production principale alors que l'autre est utilisée à des fins de contrôle et de sauvegarde. Le CNG se sert de quatre réseaux de catégorie C consécutifs, exploités par les deux réseaux internes du CNG qui sont complètement séparés (nous faisons fonctionner deux pare-feux distincts qui protègent ces deux segments de réseaux séparés). Les réseaux internes sont gérés par trois commutateurs Cisco 6000 disposant d'une capacité de routage avec une structure de base fonctionnant à 1Gbps. RIP v2 [30] est le protocole de routage interne. La configuration du matériel pare-feu de l'entreprise est standard et identique à la configuration de notre ferme de serveurs (un cluster de Linux serveurs) – un serveur DELL 2450 à double processeur avec deux Pentium III/700MHz, 1GB de RAM, un disque SCSI 18 GB. En sus du matériel de base, nous utilisons des cartes Ethernet d'1 Gbps à port double SysKonnect.

Le pare-feu du Centre fonctionne sur la version 3.3 d'OpenBSD avec des corrections stables et le noyau générique non modifié fourni au cours de l'installation du système. L'ensemble des fichiers de configuration (pour les machines individuelles) sont centralisés dans un dépôt de CVS afin de garder une trace des modifications effectuées sur la configuration du réseau (se reporter à l'annexe pour les listes complètes). En règle générale, nous éteignons tous les services/démons sur le pare-feu à l'exception de quelques uns qui ont été sélectionnés : cron(8), syslogd(8), inetd(8), getty(8), pppoe(8), gated(8) et sshd(8). Le démon ssh(8) fonctionne uniquement sur l'interface interne du pare-feu. Nous faisons également fonctionner snort(8) [31] et ntpd(8) [32, 33] afin d'avoir un IDS ou *Intrusion Detection System* de base et des estampillations dans les fichiers journal. Le serveur temps ntpd(8) est synchronisé à partir d'une source éprouvée (un serveur interne NTP).

Notre politique de sécurité met en application la technique du « refus par défaut », puis procède à l'autorisation des services entrants sélectionnés (par ex : smtp, http, ftp, telnet, ssh…) sur des machines choisies. On autorise toutes les connexions sortantes. L'ensemble des entrées interactives (telnet, shh) est concentrée sur une seule machine avec l'utilisation d'une authentification puissante grâce à l'authentification à deux facteurs RSA SecurID [34].

A l'heure actuelle, nous disposons de quatre connexions VPN permanentes (à un mélange de pairs de l'OpenBSD et de Cisco) alors que les clients nomades basés sur Windows se connectent en fonction de leurs besoins et quand ils en ont besoin. A des fins d'authentification VPN, nous utilisons la technique du secret partagé (pour des connexions permanentes et des certifications X509 pour les clients « guerriers nomades »). La stabilité du pare-feu est très élevée (temps de fonctionnement en ordre de mois) mais le système est très sensible aux coupures de réseau. A savoir qu'en raison du trafic « en arrière-plan » de l'ISAKMP (c'est-à-dire le renouvellement de la SA), nous sommes habituellement les premiers à constater tous les problèmes avec la connectivité grand réseau (c'est-à-dire perte des routes VPN). La charge du système est négligeable.

Depuis nous avons réalisé des mesures rudimentaires sur la bande passante, les résultats doivent être regardés sur le plan qualitatif et non quantitatif. Nos mesures ont été faites dans des conditions réelles, sur le réseau de production avec le trafic dû à l'activité du Centre. Le but de ces mesures n'est pas d'obtenir les meilleures performances mais de voir si le pare-feu OpenBSD présente un obstacle dans des conditions « normales ».

Nous avons réalisé trois tests utilisant ttcp[35] et iperf[36] les programmes de « benchmark ». Le premier test a été réalisé entre deux serveurs Sun E450 (tous les deux avec une carte réseau à 1 Gbps) connectés au réseau local (LAN). Le deuxième test a été réalisé entre un serveur SUN E450 (carte réseau 1Gbps) connecté au sein du réseau de production du CNG et une Sun Ultra 5 (carte réseau à 100Mbps) connectée dans le réseau de test du CNG. Le troisième test a été réalisé entre un Sun E450 (carte réseau 1Gbps) connecté dans le réseau de production du CNG et une machine Sun (carte réseau à 1Gbps) connectée dans un Institut voisin interconnecté sur REVE MAN.

Dans la première expérience, chaque paquet devait traverser l'infrastructure réseau complète du CNG pour aller d'un E450 vers l'autre à travers le LAN. Dans la deuxième expérience, chaque paquet devait traverser l'infrastructure réseau du CNG, incluant le réseau REVE MAN et les deux pare-feux OpenBSD avec chacun le filtrage de paquets et des VPNs actifs (pas de VPN entre les deux pare-feux, mais des VPN avec des pairs externes). Dans le troisième cas, les paquets devaient passer l'infrastructure complète du réseau du CNG, le pare-feu de production, le réseau métropolitain REVE MAN et l'infrastructure réseau de l'Institut voisin afin d'accéder au serveur Sun.

Le débit TCP approximatif, que nous avons obtenu entre les deux serveurs Sun dans le LAN (sans pare-feu) était entre 20-30MB/s ou 160-240Mbps. Ceci représente le trafic « normal » quand un pare-feu OpenBSD n'est pas mis en cause. Dans le second cas, les mesures du trafic TCP mesurées étaient autour de 10-12MB/s ou 80-100Mbps dans le sens serveur à 1Gbps via deux pare-feux OpenBSD vers une machine à 100Mbps. Les résultats indiquent que nous pouvons saturer le lien à 100Mbps sans aucun problème. Dans le troisième cas, le trafic maximum obtenu fut 80Mbps pour TCP et légèrement inférieur à 100Mbps pour le trafic UDP. A présent, il n'y a pas d'explication pourquoi les mesures sont si basses, mais nous étions les premiers à faire des tests de bande passante sur le MAN REVE et beaucoup de facteurs possibles peuvent expliquer ces valeurs, ceci étant étudié par le NOC REVE. Pour chaque cas intégrant les pare-feux OpenBSD, nous mesurions à l'aide de netstat(8) les valeurs pour Send-Q et Recv-Q pour l'activité réseau. Nous n'avons vu aucun paquet mis en queue indiquant que tous les paquets sont entrés et sortis du pare-feu sans délai.

3.1 Scénario N°1 : VPN de réseau privé connecté à une ligne ISDN/ADSL vers un réseau de catégorie entreprise

Ce scénario implique un travailleur à domicile isolé, connecté au réseau public d'Internet via une ligne ISDN ou ADSL qui souhaite utiliser les ressources de son réseau entreprise (voir le schéma du réseau dans la Figure n°1). La politique de sécurité est habituellement établie selon le refus par défaut pour le trafic entrant (quelquefois avec un ou deux services tels que smtp et/ou shh autorisés à entrer sur le réseau) et la transmission pour tout le trafic sortant. L'authentification VPN est basée sur le secret partagé. Actuellement, nous utilisons différentes configurations de matériel pare-feu domestique avec la spécification la plus faible qui est un ancien système IBM PS/2e avec un processeur Intel 486DX fonctionnant à 25MHz avec 16MB de RAM, un disque 300MB IDE et deux cartes Ethernet 3COM 3C509D PCMCIA (10Mbps). Cette configuration spécifique utilise le noyau générique et le chargement du système est légèrement supérieur à 1.0 avec le fonctionnement de snort(8). Sans snort(8), le chargement du système correspond approximativement à 0.2. La performance du réseau est parfaitement suffisante et les débits de transfert de données sont habituellement limités par la largeur de bande disponible ISDN/ADSL et non pas par le matériel ou le système d'exploitation. Il est important de noter que l'ISDN n'est pas une couche physique adaptée pour une connexion VPN, en raison de l'expiration de la SA et du trafic permanent d'arrière-plan qui garde la ligne téléphonique pratiquement constamment ouverte. Il est également bon de noter qu'auparavant, nous avions réussi à installer et à faire fonctionner le même système pare-feu avec seulement 4MB de RAM mais cela avait réellement exigé que le noyau soit complètement démonté (suppression de tous les « driver » superflus) et que l'on construise une disquette d'initialisation sur mesure (avec uniquement un ensemble minimum d'utilitaires) afin d'installer OpenBSD.

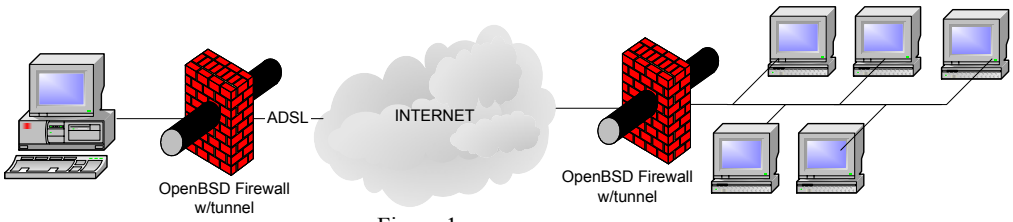

Figure 1.

3.2 Scénario N°2 : VPN de réseau du routeur Cisco vers un réseau de catégorie entreprise

Ce scénario implique un VPN de type entreprise à entreprise. Un côté de la connexion est basé sur un routeur Cisco (Cisco 4500 ; avec IOS image c4500-js56i-mz.121-3.bin) se connectant au pare-feu OpenBSD qui fait fonctionner le démon ISAKMP (se reporter au schéma du réseau de la Figure n°2). Cette configuration a été mise en place afin de permettre à une équipe de chercheurs à distance de pouvoir bénéficier d'applications spécifiques du CNG, comme si les serveurs étaient dans le réseau à distance. La mise en place de ce tunnel a été grandement facilitée par le fait que l'équipe en question possédait un réseau de classe C pour toute l'équipe, ce qui a permis très facilement de délimiter qui utilise le tunnel. Afin de mettre en place ce VPN, il a fallu que l'équipe réseau à distance, fasse une mise à jour de leur routeur Cisco afin que celui-ci puisse faire de l'IPsec et de l'échange de clefs via ISAKMPD. Une fois cette mise à jour de l'IOS 12 faite, la mise en place de la configuration côté Cisco n'a pas posé de problème (les lignes correspondantes de cette configuration sont en annexes). Au début de cette collaboration, le chiffrement du tunnel ne fut fait qu'à partir de DES et non 3DES comme c'est le cas par défaut pour les tunnels OpenBSD-OpenBSD (scénario N°1). La possibilité de faire du 3DES sur le routeur Cisco demandait un ajout de mémoire, et cette opération n'a pu être faite que plus tard. Durant ce temps, il nous a été très simple de gérer 2 protocoles de chiffrement, grâce aux fichiers isakmpd.conf(5) et isakmpd.policy(5) présents sur le pare-feu du CNG.

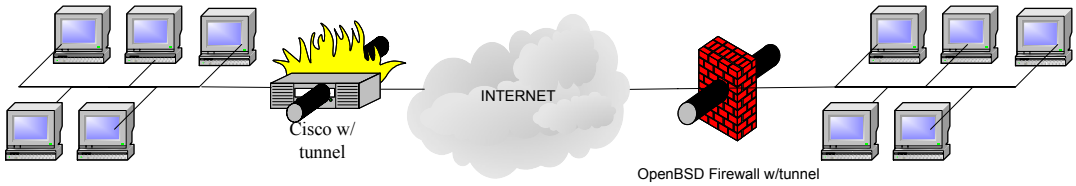

Figure 2

3.3 Scénario N°3: VPN guerrier nomade vers un réseau de catégorie entreprise

Ce scénario implique un client de type « nomade » utilisant un PC fonctionnant sur Windows 2000/XP connecté à l'Internet (avec une adresse IP dynamique) dans un cybercafé, dans un hôtel, qui vise à se servir du réseau de son entreprise (se reporter au schéma de la Figure n°3). La mise en place de cette possibilité de VPN, demande quelques changements du côté OpenBSD. Du fait que le tunnel devrait s'établir entre le réseau de l'entreprise et une adresse IP dynamiquement allouée par le fournisseur d'accès du « nomade ». Cette modification porte sur le fichier pf.conf(5), en effet les règles de filtrage doivent autoriser de n'importe où:

- le trafic encrypté (protocole ESP)
- l'échange des clefs avec ISAKMPD
- le trafic encapsulé sur l'interface « enc0 » pour le protocole ipencap
- le trafic réel, celui contenu dans le tunnel entre les 2 extrémités sur l'interface « enc0 ».

Pour le moment, les utilisateurs nomades utilisent un secret partagé pour l'authentification. Nous espérons pouvoir utiliser l'authentification X509 dans un futur proche (l'impossibilité de tracer ipsec.exe sous Windows complique un peu la chose, pour comprendre le fonctionnement de l'échange de clefs via les certificats entre OpenBSD et Windows). Sur le poste nomade, un programme ipsec.exe [37] est installé ainsi qu'un fichier de configuration ipsec.conf (voir annexe), contenant le secret partagé (les droits sur le système de fichier permettent de restreindre à son propriétaire l'accès au fichier). L'utilisateur se connecte à son fournisseur d'accès et lance ipsec.exe. Il peut alors directement utiliser le réseau de l'entreprise, le tunnel étant établi, des paquets sont envoyés vers le réseau de l'entreprise.

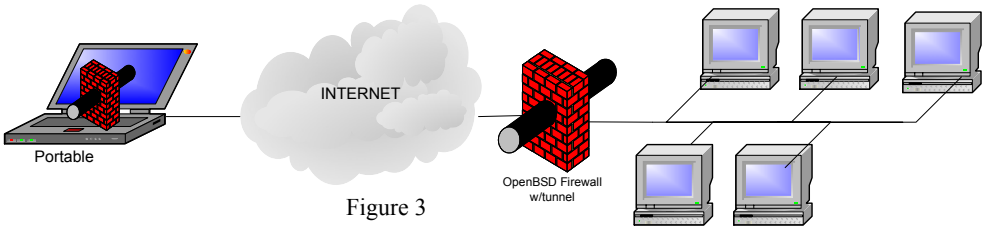

Figure 3

4 Discussion et conclusions

Puisqu'il existe une grande variété de connexions rapides et peu onéreuses à l'Internet et étant donnés les avertissements de sécurité quotidiens, il existe un réel besoin de mettre en place une solution de Réseau Privé Virtuel (VPN) et pare-feu, qui soit souple, à moindre coût et à maintenance réduite. Depuis plusieurs années, nous utilisons le système d'exploitation gratuit OpenBSD basé sur la plate-forme multiple 4.4BSD comme pare-feu et solution VPN. Le développement du système OpenBSD a commencé il y a sept ans, grâce à un audit complet d'un code source existant. Dès le début, les efforts ont porté sur la sécurité proactive et la cryptographie intégrée. La portabilité, la standardisation et l'exactitude du codage ont permis d'aboutir à des résultats globaux impressionnants, avec en plus de sept années d'existence une seule faille de sécurité dans l'installation par défaut. Dans les rares situations au cours desquelles on a détecté un problème, les corrections du code source ont été presque immédiatement disponibles, puisque la politique n'est pas (et n'a jamais été) de fournir des correctifs binaires. Cela paraît être une bonne décision puisque la reconstruction du système d'exploitation complet est très bien documentée (la documentation représente l'une des caractéristiques par laquelle se distingue véritablement le système OpenBSD) et représente une tâche relativement aisée (sur un PC de bureau type, cela prend moins de deux heures). Malgré tout cela, le système OpenBSD est toujours moins utilisé que Linux, même comme pare-feu ou solution VPN, très vraisemblablement en raison d'une publicité moindre et d'un niveau de support moins important pour des systèmes « exotiques » (ce qui n'est, de toute façon, pas particulièrement approprié à une pare-feu/passerelle de sécurité).

L'ensemble des éléments précités nous ont amenés à penser il y a quelques années déjà, que le système OpenBSD se révélait un meilleur choix pour toute une catégorie de systèmes spécifiquement adaptés au fonctionnement des réseaux : les pare-feux et les passerelles de sécurité en particulier. L'introduction du démon ISAKMP lui-même, sans parler de l'utilisation du système OpenBSD comme plate-forme de référence dans le contrôle de l'interopérabilité VPN, fait de lui un candidat idéal pour la mise en place d'une solution VPN à moindre coût et à gestion réduite. Grâce à la souplesse du dispositif de filtrage de paquets intégré « Stateful Packet Filter », nous avons développé un ensemble de fichiers de configuration pare-feu génériques. Nous avons présenté trois cas différents d'utilisation au Centre, qui sont au niveau entreprise, avec une passerelle de sécurité pare-feu basée sur le système OpenBSD disposant d'une connexion multi-port/multi-gigabit. Nous avons démontré qu'il est possible de ré-utiliser le matériel totalement obsolète - matériel lançant le système OpenBSD - comme pare-feu « dual-homed » (via une ligne ADSL) fonctionnant de manière plus qu'adéquate. Nous avons développé des configurations VPN du système OpenBSD qui supportent tout aussi bien le type de connexions VPN statique que le type « guerrier nomade » sur différentes passerelles de sécurité telles qu'OpenBSD, Cisco et Windows. Nous avons constaté que le système OpenBSD était extrêmement performant, résistant, facile à installer, à configurer et à utiliser, avec pratiquement aucune exigence de maintenance.

Annexe

Tous les fichiers de configuration pour les pare-feux/VPN discutés dans cet article (aussi bien que l'article lui-même) sont disponible sur le site web du CNG à : http://software.cng.fr/vpn/

Remerciements

Les auteurs tiennent à remercier Claude Scarpelli (Genoscope - Centre National de Séquençage), Jehan Procaccia (Institut National des Télécommunications) et Béatrice Bourgeois (Centre National de Génotypage) pour leur contribution aux tests réalisés dans le cadre de cet article.

Références

1. Gramp, F. and R.H. Morris, UNIX operating system security. *AT&T Bell Labs Technical Journal*. **63**(8, Part 2): 1649-1672, 1984.
2. *The OpenBSD Project*, http://www.openbsd.org.
3. Seifried, K., *Why Linux will never be as secure as OpenBSD*, http://www.seifried.org/security/os/20011107-linux-openbsd.html.
4. Seifried, K., *Why OpenBSD will never be as secure as Linux*, http://www.seifried.org/security/os/20011107-openbsd-linux.html.
5. *Immunix*, http://www.immunix.org/.
6. *Security-Enhanced Linux*, http://www.nsa.gov/selinux/.
7. *Openwall Project*, http://www.openwall.com/.

8. *The OpenSSL Project*, http://www.openssl.org/.
9. *The OpenSSH Project*, http://www.openssh.com/.
10. Gilmore, J., *The FreeS/Wan Project*, http://www.freeswan.org/.
11. *NetSec*, http://www.netsec.net/.
12. *Broadcom Corporation*, http://www.broadcom.com/products/5802.html.
13. *Soekris Engineering*, http://www.soekris.com/vpn1201.htm.
14. *SafeNet SafeXcel*, http://www.ashleylaurent.com/products/chipsets/safenet.htm.
15. Reed, D., *IP Filter - TCP/IP Firewall/NAT Software*, http://coombs.anu.edu.au/ipfilter/.
16. Hartmeier, D., *Design and Performance of the OpenBSD Stateful Packet Filter (pf)*, http://www.benzedrine.cx/pf-paper.html.
17. van Rooij, G. Real Stateful TCP Packet Filtering in IP Filter. in *2nd International SANE Conference*, Maastricht, Netherlands, 2000.
18. Fergusson, P. and G. Huston, What is a VPN? - Part I. *The Internet Protocol Journal*. **1**(1): 2-19, 1998.
19. Fergusson, P. and G. Huston, What is a VPN? - Part II. *The Internet Protocol Journal*. **1**(2): 2-18, 1998.
20. Ioannidis, J. and M. Blaze. The Architecture and Implementation of Network-Layer Security Under Unix. in *Proceedings of the Fourth Usenix Security Symposium*. 29-39, Santa Clara, CA, 1993.
21. Kent, S. and R. Atkinson, Security Architecture for the Internet Protocol. *RFC 2401*. 1998.
22. Hallqvist, N. and A.D. Keromytis. Implementing Internet Key Exchange (IKE). in *Proceedings of the USENIX Annual Technical Conference, Freenix Track*. 201-214, San Diego, CA, 2000.
23. Kent, S. and R. Atkinson, IP Encapsulating Security Payload (ESP). *RFC 2406*. 1998.
24. Kent, S. and R. Atkinson, IP Authentication Header. *RFC 2402*. 1998.
25. Harkins, D. and D. Carrel, The Internet Key Exchange (IKE). *RFC 2409*. 1998.
26. Orman, H., The OAKLEY Key Determination Protocol. *RFC 2412*. 1998.
27. Diffie, W. and M.E. Hellman, New directions in cryptography. *IEEE Transactions on Information Theory*. **IT-22**(6): 644-654, 1976.
28. Orrado, G. and C. Scarpelli. REVE, Réseau d'Evry Val d'Essonne. in *Actes du 4éme Journées Réseaux JRES2001*. 577-584, Lyon, 2001.
29. Rekhter, Y. and Y. Li, A Border Gateway Protocol 4 (BGP-4). *RFC 1771*. 1995.
30. Malkin, G., RIP Version 2. *RFC 2453*. 1998.
31. *Snort - The Open Source Network Intrusion Detection System*, http://www.snort.org/.
32. Mills, D.L., *ntpd - Network Time Protocol (NTP) daemon*, http://www.eecis.udel.edu/~mills/ntp/html/ntpd.html.
33. Mills, D.L., Network Time Protocol (Version 3) Specification, Implementation and Analysis. *RFC 1305*. 1992.
34. *RSA SecurID*, http://www.rsasecurity.com/products/securid/.
35. Slattery, T.C. and M. Muuss, *TTCP: Test TCP connection*, http://www.netcordia.com/tools/tools/TTCP/ttcp.html.
36. Gates, M., A. Warshavsky, and J. Pietsch, *Iperf: The TCP/UDP Bandwidth Measurement Tool*, http://dast.nlanr.net/Projects/Iperf/.
37. Müller, M., *Windows 2000/XP VPN Tool*, http://vpn.ebootis.de/.